

Guide d'utilisation de *LARP*

Troisième édition

Marco Lavoie

Copyright © 2004-2008 Marco Lavoie

Guide d'utilisation de LARP

Troisième édition

Imprimé le 2008-02-12

Marco Lavoie

Copyright © 2004-2008 Marco Lavoie

Tous droits réservés.

Toute reproduction, en tout ou en partie, sous quelque forme et par quelque procédé que ce soit, est interdite sans l'autorisation écrite préalable de l'auteur.

Les images, pseudo-codes, organigrammes, copies d'écrans et fenêtres présentés dans cet ouvrage le sont à titre d'illustration et peuvent être différents dans la réalité. En conséquence, les procédures et manipulations décrites ainsi que cet ouvrage sont publiés en l'état, et l'utilisateur supporte tous les risques en ce qui concerne leur qualité et fonctionnement.

Tous les logiciels, ordinateurs et marques cités dans cet ouvrage sont des marques de commerce déposées et ne sont cités qu'à titre d'exemples.

Bien que l'auteur ait testé les pseudo-codes, organigrammes, procédures et manipulations décrites dans cet ouvrage et revu leur contenu, l'auteur n'offre aucune garanties, expresse ou tacite, concernant l'ouvrage ou son contenu.

L'auteur a fait tout ce qui était en son pouvoir pour retrouver les copyrights et marques de commerce. On peut lui signaler tout renseignement menant à la correction d'erreurs ou d'omissions.

Guide d'utilisation de *LARP*

Table des matières

1	Introduction.....	17
1.1	Licence d'utilisation	18
1.1.1	Version gratuitiel.....	19
1.1.2	Version partagiciel.....	19
1.1.3	Licence	19
1.2	Auteur de <i>LARP</i>	21
1.3	Installation	21
1.3.1	Exigences minimales d'équipements et logiciels	21
1.3.2	Installation à partir d'un CD	22
1.3.3	Installation à partir d'un fichier téléchargé.....	22
1.3.4	Désinstallation	23
1.4	Enregistrement.....	23
1.4.1	Procédure d'enregistrement	24
1.4.2	Commander des clés de débridage	25
1.5	Mises à jour de <i>LARP</i>	26
1.6	Support technique	27
1.6.1	Aide en ligne.....	27
1.6.2	Rapporter un bogue	28
1.6.3	Site Web de <i>LARP</i>	30
2	Environnement de développement.....	33
2.1	Aide disponible dans <i>LARP</i>	33
2.2	Éléments de l'interface	33
2.2.1	Barre de menu.....	35
2.2.2	Panneau de contrôle	39
2.2.3	Navigateur de documents	41
2.2.4	Panneau de modèles	41
2.2.5	Éditeurs	42
2.2.6	Panneau de messages.....	44
2.2.7	Panneau de statut	44
2.2.8	Console d'exécution	45
2.2.9	Fenêtre d'exécution pas-à-pas.....	46
2.3	Fonctionnalités de l'éditeur textuel.....	47
2.3.1	Éditer un document textuel.....	47
2.3.2	Recherche et remplacement	48
2.3.3	Surligne de contenu	49
2.3.4	Configuration de l'éditeur textuel.....	49
2.3.5	Commandes d'édition de l'éditeur textuel	49
2.3.5.1	Commandes de l'éditeur textuel accessibles via les menus	49
2.3.5.2	Commandes de l'éditeur textuel accessibles via le clavier	50
2.3.5.3	Contrôle de l'éditeur textuel via la souris.....	51
2.4	Fonctionnalités de l'éditeur graphique	51
2.4.1	Instructions d'organigramme	52
2.4.2	Éditer un organigramme.....	53
2.4.3	Manipulation d'instructions d'organigramme	54
2.4.3.1	Insertion, déplacement et destruction d'instructions d'organigramme	55
2.4.3.2	Édition d'instructions d'organigramme.....	56
2.4.4	Recherche et remplacement dans un organigramme	57
2.4.5	Agrandissement de l'affichage	57

2.4.6	Surligne d'instructions en exécution pas-à-pas	57
2.4.7	Configuration de l'éditeur graphique	58
2.4.8	Commandes d'édition de l'éditeur graphique	58
2.4.8.1	Commandes de l'éditeur graphique accessibles via les menus.....	58
2.4.8.2	Commandes de l'éditeur graphique accessibles via le clavier	59
2.4.8.3	Contrôle de l'éditeur graphique via la souris	60
2.5	Compilation et exécution.....	60
2.5.1	Exécution d'un projet.....	61
2.5.2	Exécution pas-à-pas.....	61
2.5.2.1	Interface de l'exécution pas-à-pas.....	61
2.5.2.2	Modes d'exécution pas-à-pas.....	64
2.5.2.3	Inspection des variables	65
2.5.2.4	Inspection de la pile d'appels	66
2.5.2.5	Points d'arrêt.....	66
2.5.2.6	Animation	68
2.5.3	Sauvegarde de sécurité	70
2.5.4	Avertissements et erreurs	70
2.6	Configuration de <i>LARP</i>	71
2.6.1	Configuration générale.....	71
2.6.1.1	Configuration des éditeurs.....	72
2.6.1.2	Configuration de la console d'exécution.....	73
2.6.1.3	Configuration du mode super-utilisateur et du système de mises à jour	74
2.6.2	Sélection de couleurs	76
2.6.2.1	Couleurs dans la console d'exécution	76
2.6.2.2	Couleurs dans les éditeurs	77
2.6.2.3	Couleurs pour exécution pas-à-pas.....	77
3	Mode super-utilisateur	79
3.1	Prévention du plagiat	79
3.1.1	Pseudonyme actif.....	80
3.1.2	Pseudonyme rattaché aux fichiers projet.....	81
3.1.3	Chiffrement des documents	81
3.1.4	Contrôle du copier-coller	82
3.1.5	Contrôle de l'impression.....	83
3.2	Débridage de l'environnement de développement	83
3.2.1	Sélection d'une technologie de clé.....	84
3.2.2	Statistiques de projet.....	85
3.2.3	Conversion d'organigramme en pseudo-code	85
3.2.4	Projets publics	86
4	Un premier algorithme.....	89
4.1	Les commentaires	89
4.2	Début et fin d'un algorithme	90
4.3	Syntaxe des instructions	90
4.4	Séparation des instructions.....	91
4.5	Création du projet <i>LARP</i>	92
5	Constantes et variables	97
5.1	Noms de variables	97
5.2	Opérations.....	98
5.3	Valeurs numériques	99
5.4	Les chaînes de caractères	99
5.5	Séquences d'échappement.....	100
5.6	L'affectation	100
6	Conteneurs.....	101

6.1	Regroupement de valeurs	101
6.2	Accès aux éléments	101
6.3	Retirer des éléments	102
7	Lecture et écriture	105
7.1	Instruction d'entrées/sorties pour organigrammes	105
7.2	L'instruction de lecture	106
7.3	L'instruction d'écriture	107
7.4	L'instruction de requête	108
7.5	Le séparateur	109
8	Opérateurs et fonctions prédéfinies	113
8.1	Opérateurs arithmétiques	113
8.2	Opérateurs de chaînes de caractères	114
8.3	Opérateurs de conteneurs	115
8.4	Fonctions prédéfinies	116
8.4.1	Fonctions mathématiques	116
8.4.2	Fonctions de manipulation de chaînes	117
8.4.3	Fonctions de manipulation de conteneurs	118
9	Structures conditionnelles	121
9.1	Les conditions	122
9.2	Opérateurs relationnels	122
9.3	Tests de type	123
9.4	Opérateurs logiques	124
9.5	Priorité des opérateurs	125
9.6	Structures SI et SI-SINON	126
9.7	Structures SI-SINON imbriquées	128
9.8	Structure SI-SINON-SI	130
9.9	Structure de sélection	134
10	Structures répétitives	137
10.1	Structure TANTQUE	137
10.2	Structure RÉPÉTER-JUSQU'À	139
10.3	Structure POUR	141
11	Modules	145
11.1	Noms de modules	145
11.2	Module principal	146
11.3	Modules auxiliaires	147
11.4	Variables locales	149
11.5	Modules auxiliaires avec paramètres	150
11.5.1	Déclaration des paramètres d'un module	150
11.5.2	Paramètres valeurs	152
11.5.3	Paramètres références	153
11.6	Modules avec valeur de retour	155
11.7	Syntaxe d'invocation alternative	157
12	Fichiers et tampons d'entrées/sorties	159
12.1	Tampons d'entrées/sorties	159
12.2	Fichiers	160
12.3	Canaux d'entrées/sorties	161
12.3.1	Ouverture d'un document	161
12.3.2	Ouvrir un tampon d'entrées/sorties	161
12.3.3	Ouvrir un fichier	162
12.3.4	Modes d'accès	163

12.3.5	Fermeture d'un canal d'entrées/sorties	164
12.4	Lecture via un canal d'entrées/sorties	164
12.5	Écriture via un canal d'entrées/sorties	165
12.6	Détection de fin de contenu via un canal d'entrées/sorties	166
Annexe A - Calcul binaire.....		167
A.1	Pourquoi les ordinateurs sont-ils binaires?	167
A.2	La numérotation en base décimale.....	167
A.3	La numérotation en base binaire	168
A.4	La numérotation hexadécimale.....	170
Annexe B - Récursivité.....		173
Annexe C - Fonctions prédéfinies.....		175
	ABSOLU	175
	ALÉATOIRE	176
	ARCTANGENTE	177
	ARRONDIR	178
	CAPACITÉ	179
	COMPTER	179
	COSINUS	180
	DATE	181
	ENCARACTÈRES.....	182
	ENCHAÎNE.....	183
	EXP	183
	FINDECONTENU.....	184
	FORMATER	185
	HEURE	189
	LOG10.....	189
	LOGE.....	190
	MAJUSCULES	191
	MAXIMUM	191
	MINIMUM	192
	MINUSCULES.....	193
	PI	194
	PLAFOND	194
	PLANCHER.....	195
	POSITION	196
	RACINE.....	197
	SINUS.....	197
	SOUSENSEMBLE.....	198
Annexe D - Éléments de syntaxe.....		201
D.1	Modules.....	201
D.2	Structures conditionnelles.....	201
D.3	Structures répétitives	203
D.4	Fichiers et tampons d'entrées/sorties	204
D.5	Lecture et écriture	205
Annexe E - Avertissements et erreurs.....		207
E.1	Messages reliés à l'environnement de développement.....	207
E.2	Messages reliés à l'exécution d'algorithmes.....	210
Index		225

Guide d'utilisation de *LARP*

Liste des figures

Figure 1-1 : Identification de la version gratuitiel	19
Figure 1-2 : Identification de la version partagiciel	19
Figure 1-3 : Démarrage de <i>LARP</i>	24
Figure 1-4 : Enregistrement de <i>LARP</i>	25
Figure 1-5 : Clé de débridage	25
Figure 1-6 : Fenêtre d'accueil	25
Figure 1-7 : Téléchargement et installation des mises à jour.....	26
Figure 1-8 : Messages d'erreur.....	28
Figure 1-9 : Erreur d'application	28
Figure 1-10 : Rapport de bogue (formulaire)	29
Figure 1-11 : Rapport de bogue (fichiers)	30
Figure 1-12 : Rapport de bogue (transmission).....	31
Figure 2-1 : Fenêtre principale	34
Figure 2-2 : Panneau de contrôle	39
Figure 2-3 : Navigateur de documents	41
Figure 2-4 : Modèles pour pseudo-code	42
Figure 2-5 : Modèles pour organigramme	42
Figure 2-6 : Éditeur textuel de <i>LARP</i>	43
Figure 2-7 : Éditeur graphique de <i>LARP</i>	43
Figure 2-8 : Panneau de messages	44
Figure 2-9 : Panneau de statut	44
Figure 2-10 : Console d'exécution	45
Figure 2-11 : Fenêtre d'exécution pas-à-pas.....	46
Figure 2-12 : Panneau de l'éditeur textuel.....	47
Figure 2-13 : Fenêtre de recherche	48
Figure 2-14 : Fenêtre de recherche et remplacement.....	48
Figure 2-15 : Panneau de l'éditeur graphique	52
Figure 2-16 : Nouveau module principal	54
Figure 2-17 : Nouveau module auxiliaire.....	54
Figure 2-18 : Nœud d'insertion sélectionné.....	54
Figure 2-19 : Instruction d'organigramme sélectionnée	54
Figure 2-20 : Insertion d'une instruction d'organigramme via le menu contextuel du noeud	56
Figure 2-21 : Édition d'une instruction d'organigramme	57
Figure 2-22 : Prochaine instruction à exécuter.....	58
Figure 2-23 : Point d'arrêt surligné	58
Figure 2-23 : Fenêtre de compilation	61
Figure 2-24 : Éléments de la fenêtre d'exécution pas-à-pas.....	62
Figure 2-25 : Panneau de statut de la fenêtre d'exécution pas-à-pas	63
Figure 2-26 : Prochaine instruction à exécuter (pas-à-pas) dans l'éditeur textuel	64
Figure 2-27 : Prochaine instruction à exécuter (pas-à-pas) dans l'éditeur graphique	65
Figure 2-28 : Panneau d'inspection des variables	65
Figure 2-29 : Panneau d'inspection de la pile d'appels	66
Figure 2-30 : Panneau d'inspection des points d'arrêt	67
Figure 2-31 : Point d'arrêt surligné dans l'éditeur textuel	67
Figure 2-32 : Point d'arrêt surligné dans l'éditeur graphique.....	68
Figure 2-33 : Panneau d'animation d'instruction	69
Figure 2-34 : Animation de l'évaluation d'une condition	69
Figure 2-35 : Configuration générale.....	72
Figure 2-36 : Configuration de l'exécution d'algorithmes	73
Figure 2-37 : Sélection du répertoire pour fichiers temporaires	74

Figure 2-38 : Configuration du mode super-utilisateur et du système de mise à jour.....	75
Figure 2-39 : Test de débridage	75
Figure 2-40 : Configuration des couleurs	76
Figure 3-1 : Spécifier le pseudonyme actif	80
Figure 3-2 : Impression des documents	83
Figure 3-3 : Clé de débridage (port USB).....	83
Figure 3-4 : Clé de débridage (port parallèle)	84
Figure 3-5 : Indicateur du mode super-utilisateur.....	84
Figure 3-6 : Statistiques de projet.....	85
Figure 3-7 : Conversion d'un organigramme en pseudo-code	86
Figure 4-1 : Modèles de commentaire.....	90
Figure 4-2 : Fenêtre d'accueil.....	92
Figure 4-3 : Fenêtre d'enregistrement	93
Figure 4-4 : Pseudonyme d'utilisateur	93
Figure 4-5 : Fenêtre Nouveau	94
Figure 4-6 : Projet <i>LARP</i> en cours d'édition	95
Figure 4-7 : Exécution du projet	95
Figure 4-8 : Messages générés lors de la compilation et l'exécution.....	96
Figure 4-9 : Sauvegarder le projet.....	96
Figure 7-1 : Instruction d'entrées/sorties pour organigrammes.....	105
Figure 7-2 : Édition d'une instruction d'entrées/sorties d'organigramme	105
Figure 7-3 : Interprétation du texte fourni lors d'une lecture	106
Figure 7-4 : Lecture de valeurs dans un organigramme	107
Figure 7-5 : Instruction d'écriture	107
Figure 7-6 : Écriture de valeurs dans un organigramme.....	108
Figure 7-7 : Interrogation peu élégante	108
Figure 7-8 : Instruction REQUÊTE	109
Figure 7-9 : Formuler une requête dans un organigramme	109
Figure 7-10 : Changement de séparateur	110
Figure 7-11 : Lecture exploitant le séparateur.....	110
Figure 9-1 : Condition composée	126
Figure 9-2 : Ordre d'évaluation de la condition précédente	126
Figure 9-3 : Représentation graphique de l'ordre d'évaluation	126
Figure 11-1 : Création d'un module auxiliaire.....	146
Figure 11-2 : Affichage lors de l'exécution du module auxiliaire Menu	148
Figure 11-3 : Transfert de données via paramètres	150
Figure 12-1 : Tampons d'entrées/sorties d'un projet.....	160
Figure 12-2 : Spécifier un numéro de canal dans une instruction de lecture d'organigramme...	165
Figure 12-3 : Spécifier un numéro de canal dans une instruction d'écriture d'organigramme....	166
Figure C-1 : Résultats de l'invocation de ABSOLU	176
Figure C-2 : Résultats de l'invocation de ALÉATOIRE	177
Figure C-3 : Résultats de l'invocation de ARCTANGENTE	178
Figure C-4 : Résultats de l'invocation de ARRONDIR	178
Figure C-5 : Résultats de l'invocation de CAPACITÉ.....	179
Figure C-6 : Résultats de l'invocation de COMPTER.....	180
Figure C-7 : Résultats de l'invocation de COSINUS	181
Figure C-8 : Résultats de l'invocation de DATE	181
Figure C-9 : Résultats de l'invocation de ENCARACTÈRES	182
Figure C-10 : Résultats de l'invocation de ENCHAÎNE	183
Figure C-11 : Résultats de l'invocation de EXP.....	184
Figure C-12 : Résultats de l'invocation de FORMATER	188
Figure C-13 : Résultats de l'invocation de HEURE	189

Figure C-14 : Résultats de l'invocation de LOG10	190
Figure C-15 : Résultats de l'invocation de LOGE.....	190
Figure C-16 : Résultats de l'invocation de MAJUSCULES	191
Figure C-17 : Résultats de l'invocation de MAXIMUM	192
Figure C-18 : Résultats de l'invocation de MINIMUM	193
Figure C-19 : Résultats de l'invocation de MINUSCULES	193
Figure C-20 : Résultats de l'invocation de PI	194
Figure C-21 : Résultats de l'invocation de PLAFOND	195
Figure C-22 : Résultats de l'invocation de PLANCHER	195
Figure C-23 : Résultats de l'invocation de POSITION	196
Figure C-24 : Résultats de l'invocation de RACINE	197
Figure C-25 : Résultats de l'invocation de SINUS.....	198
Figure C-26 : Résultats de l'invocation de SOUSENSEMBLE	199

Guide d'utilisation de *LARP*

Liste des tableaux

Tableau 2-1 : Commandes de la barre de menu.....	39
Tableau 2-2 : Éléments d'interface du panneau de contrôle.....	41
Tableau 2-3 : Commandes de l'éditeur textuel accessibles via le clavier.....	51
Tableau 2-4 : Instructions d'organigramme.....	53
Tableau 2-5 : Commandes de l'éditeur graphique accessibles via le clavier	59
Tableau 5-1 : Séquences d'échappement.....	100
Tableau 6-1 : Fonctions de manipulation de conteneurs	103
Tableau 7-1 : Interprétation des valeurs lues	106
Tableau 7-2 : Interprétation des espaces par LIRE et ÉCRIRE.....	110
Tableau 8-1 : Opérateurs arithmétiques.....	113
Tableau 8-3 : Opérateurs de chaînes de caractères.....	114
Tableau 8-4 : Opérateurs de conteneur	115
Tableau 8-5 : Fonctions mathématiques prédéfinies	117
Tableau 8-6 : Fonctions prédéfinies de manipulation de chaînes de caractères.....	118
Tableau 8-7 : Fonctions prédéfinies de manipulation de conteneurs.....	119
Tableau 9-1 : La structure conditionnelle SI.....	121
Tableau 9-2 : Opérateurs relationnels.....	122
Tableau 9-3 : Opérateurs logiques.....	125
Tableau 9-4 : Priorité des opérateurs.....	125
Tableau 9-5 : Structures conditionnelles SI et SI-SINON.....	126
Tableau 9-6 : Structure conditionnelle SI-SINON-SI.....	132
Tableau 9-7 : Instructions d'organigrammes requises dans une structure SI-SINON-SI.....	133
Tableau 9-8 : Structure de sélection.....	135
Tableau 9-9 : Instructions d'organigrammes requises dans une structure de sélection	136
Tableau 10-1 : Structure répétitive TANTQUE.....	137
Tableau 10-2 : Structure répétitive RÉPÉTER-JUSQU'À	139
Tableau 10-3 : Structure répétitive POUR.....	141
Tableau 12-1 : Table des codes ASCII	169
Tableau 12-2 : Correspondance des bases de numérotation	170

Guide d'utilisation de *LARP*

Liste des pseudo-codes

Pseudo-code 1-1 : Un pseudo-code.....	17
Pseudo-code 4-1 : Pseudo-code tout bête.....	89
Pseudo-code 4-2 : Les commentaires.....	89
Pseudo-code 4-3 : Syntaxe des instructions.....	90
Pseudo-code 4-4 : Séparation des instructions.....	91
Pseudo-code 5-1 : Noms de variables.....	98
Pseudo-code 5-2 : Les opérateurs.....	98
Pseudo-code 5-3 : Les valeurs numériques.....	99
Pseudo-code 5-4 : La chaîne de caractères.....	99
Pseudo-code 5-5 : L'apostrophe et le guillemet.....	99
Pseudo-code 5-6 : L'affectation.....	100
Pseudo-code 6-1 : Les conteneurs.....	101
Pseudo-code 6-2 : Conteneur dans un conteneur.....	101
Pseudo-code 6-3 : Accès aux éléments d'un conteneur.....	101
Pseudo-code 6-4 : Accès aux éléments d'un conteneur.....	102
Pseudo-code 6-5 : Éléments indéfinis d'un conteneur.....	102
Pseudo-code 7-1 : Lecture d'une valeur.....	106
Pseudo-code 7-2 : Lecture de plusieurs valeurs.....	106
Pseudo-code 7-3 : L'instruction d'écriture.....	107
Pseudo-code 7-4 : Interroger l'utilisateur.....	108
Pseudo-code 7-5 : Instruction REQUÊTE.....	108
Pseudo-code 7-6 : Une requête pour lire plus d'une valeur.....	109
Pseudo-code 7-7 : Instruction SÉPARATEUR.....	110
Pseudo-code 7-8 : Modification du séparateur.....	110
Pseudo-code 8-1 : Les opérateurs arithmétiques.....	113
Pseudo-code 8-2 : L'opérateur + appliqué à des valeurs de type distincts.....	114
Pseudo-code 8-3 : Joindre des conteneurs.....	115
Pseudo-code 8-4 : Soustraction de conteneurs.....	115
Pseudo-code 9-1 : Les conditions simples.....	123
Pseudo-code 9-2 : Condition vérifiant si une valeur est définie ou non.....	124
Pseudo-code 9-3 : Les conditions composées.....	125
Pseudo-code 9-4 : Structures SI et SI-SINON.....	127
Pseudo-code 9-5 : Syntaxes invalides de la structure SI.....	128
Pseudo-code 9-6 : Séquence de structures SI reliées.....	128
Pseudo-code 9-7 : Structures SI-SINON imbriquées les unes dans les autres.....	129
Pseudo-code 9-8 : Structures conditionnelles imbriquées.....	130
Pseudo-code 9-9 : Structure conditionnelle SI-SINON-SI.....	133
Pseudo-code 9-10 : Structure SI-SINON-SI pour tester une même valeur.....	134
Pseudo-code 9-11 : Structure de sélection.....	134
Pseudo-code 10-1 : Structure répétitive TANTQUE.....	138
Pseudo-code 10-2 : Structure répétitive RÉPÉTER-JUSQU'À.....	140
Pseudo-code 10-3 : Structure TANTQUE équivalente à la boucle du Pseudo-code 10-2.....	140
Pseudo-code 10-4 : Itérer d'une valeur à la suivante avec TANTQUE.....	142
Pseudo-code 10-5 : Structure répétitive POUR.....	142
Pseudo-code 10-6 : Structure POUR avec incrément autre que 1.....	143
Pseudo-code 10-7 : Structure répétitive POUR avec décrétement.....	143
Pseudo-code 10-8 : Structure répétitive POUR avec décrétement autre que -1.....	144

Pseudo-code 10-9 : Structure POUR invalide	144
Pseudo-code 11-1 : Le module principal	147
Pseudo-code 11-2 : Module auxiliaire	147
Pseudo-code 11-3 : Invocation d'un module auxiliaire	148
Pseudo-code 11-4 : Variables locales	149
Pseudo-code 11-5 : Module auxiliaire avec paramètres	150
Pseudo-code 11-6 : Invocation d'un module auxiliaire avec arguments	151
Pseudo-code 11-7 : Déclaration de paramètres références	152
Pseudo-code 11-8 : Passage d'arguments par valeur	153
Pseudo-code 11-9 : Déclaration de paramètres valeurs et références	154
Pseudo-code 11-10 : Module principal	154
Pseudo-code 11-11 : Module de sélection d'une commande de menu	156
Pseudo-code 11-12 : Invocation d'un module avec valeur de retour	157
Pseudo-code 11-13 : Syntaxe alternative d'invocation d'un module	157
Pseudo-code 11-14 : Invocation d'une fonction prédéfinie	158
Pseudo-code 12-1 : Ouvrir un tampon d'entrées/sorties	161
Pseudo-code 12-2 : Instructions OUVRIR invalides	162
Pseudo-code 12-3 : Ouvrir un fichier	162
Pseudo-code 12-4 : Fermer un tampon d'entrées/sorties ou un fichier	164
Pseudo-code 12-5 : Lecture d'un document via un canal d'entrées/sorties	164
Pseudo-code 12-6 : Détection de la fin d'un document lors d'une lecture	166
Pseudo-code B-1 : Calcul de la factorielle (module récursif)	173
Pseudo-code B-2 : Calcul de la factorielle (module itératif)	174
Pseudo-code C-1 : Exemples d'invocation de ABSOLU	175
Pseudo-code C-2 : Exemples d'invocation de ALÉATOIRE	176
Pseudo-code C-3 : Exemples d'invocation de ARCTANGENTE	177
Pseudo-code C-4 : Exemples d'invocation de ARRONDIR	178
Pseudo-code C-5 : Exemples d'invocation de CAPACITÉ	179
Pseudo-code C-6 : Exemples d'invocation de COMPTER	180
Pseudo-code C-7 : Exemples d'invocation de COSINUS	180
Pseudo-code C-8 : Exemples d'invocation de DATE	181
Pseudo-code C-9 : Exemples d'invocation de ENCARACTÈRES	182
Pseudo-code C-10 : Exemples d'invocation de ENCHAÎNE	183
Pseudo-code C-11 : Exemples d'invocation de EXP	184
Pseudo-code C-12 : Exemples d'invocation de FINDECONTENU	184
Pseudo-code C-13 : Exemples d'invocation de FORMATER	188
Pseudo-code C-14 : Exemples d'invocation de HEURE	189
Pseudo-code C-15 : Exemples d'invocation de LOG10	190
Pseudo-code C-16 : Exemples d'invocation de LOGE	190
Pseudo-code C-17 : Exemples d'invocation de MAJUSCULES	191
Pseudo-code C-18 : Exemples d'invocation de MAXIMUM	192
Pseudo-code C-19 : Exemples d'invocation de MINIMUM	192
Pseudo-code C-20 : Exemples d'invocation de MINUSCULES	193
Pseudo-code C-21 : Exemples d'invocation de PI	194
Pseudo-code C-22 : Exemples d'invocation de PLAFOND	195
Pseudo-code C-23 : Exemples d'invocation de PLANCHER	195
Pseudo-code C-24 : Exemples d'invocation de POSITION	196
Pseudo-code C-25 : Exemples d'invocation de RACINE	197
Pseudo-code C-26 : Exemples d'invocation de SINUS	198
Pseudo-code C-27 : Exemples d'invocation de SOUSENSEMBLE	198

Guide d'utilisation de *LARP*

Liste des organigrammes

Organigramme 1-1 : Un organigramme	18
Organigramme 4-1 : Organigramme tout bête	89
Organigramme 4-2 : Syntaxe des instructions	91
Organigramme 4-3 : Prolongation d'une instruction d'organigramme.....	92
Organigramme 5-1 : Opérations séquentielles.....	99
Organigramme 5-2 : Exemple de séquences d'échappement	100
Organigramme 6-1 : Modifier les éléments d'un conteneur	102
Organigramme 6-2 : Compresser un conteneur.....	103
Organigramme 7-1 : Changement de séparateur	111
Organigramme 8-1 : La concaténation de chaînes de caractères	114
Organigramme 8-2 : Test de type CONTENEUR.....	116
Organigramme 9-1 : Condition testant le type du contenu d'une variable	124
Organigramme 9-2 : Structure SI	127
Organigramme 9-3 : Structure SI-SINON.....	127
Organigramme 9-4 : Structures conditionnelles imbriquées	131
Organigramme 9-5 : Structure conditionnelle SI-SINON-SI.....	133
Organigramme 9-6 : Exemple de structure de sélection	135
Organigramme 10-1 : Structure répétitive TANTQUE.....	138
Organigramme 10-2 : Structure répétitive RÉPÉTER-JUSQU'À	140
Organigramme 10-3 : Structure répétitive POUR.....	143
Organigramme 11-1 : Le module principal	147
Organigramme 11-2 : Module auxiliaire simple sous forme d'organigramme.....	148
Organigramme 11-3 : Organigramme invoquant un module auxiliaire	149
Organigramme 11-4 : Module auxiliaire avec paramètres	151
Organigramme 11-5 : Invocation d'un module auxiliaire avec arguments	151
Organigramme 11-6 : Déclaration de paramètres références.....	152
Organigramme 11-7 : Passage d'arguments par valeur	153
Organigramme 11-8 : Déclaration de paramètres références.....	155
Organigramme 11-9 : Module auxiliaire avec valeur de retour	156
Organigramme 11-10 : Deux exemples d'invocation d'un même module.....	158
Organigramme 12-1 : Exploitation du mot réservé TAMPON	162
Organigramme 12-2 : Spécification du mode d'accès à un document.....	163
Organigramme 12-3 : Fermer plusieurs canaux d'entrées/sorties	164
Organigramme 12-4 : Écriture dans un document via un canal d'entrées/sorties	165

1 Introduction

LARP est en fait un acronyme. Il vient de la compression de la phrase «*Logiciel d'Algorithmes et de Résolution de Problèmes*», conçu par [Marco Lavoie](#). *LARP* est un langage de programmation permettant le prototypage rapide d'algorithmes.

L'avantage de *LARP* est que le programme est un langage pseudo-code à syntaxe flexible et non un code source à compiler, ce qui permet de formuler des algorithmes en un langage semi-naturel plutôt que de devoir adhérer à une syntaxe rigide et cryptique telle que celle des langages de programmation traditionnels (C++, Pascal, Java, etc.).

Voici un pseudo-code *LARP* indiquant à l'écran si un nombre entré via le clavier de l'ordinateur est positif ou négatif :

```
\\ Pseudo-code simple
DÉBUT
  ÉCRIRE "Entrez un nombre"
  LIRE N

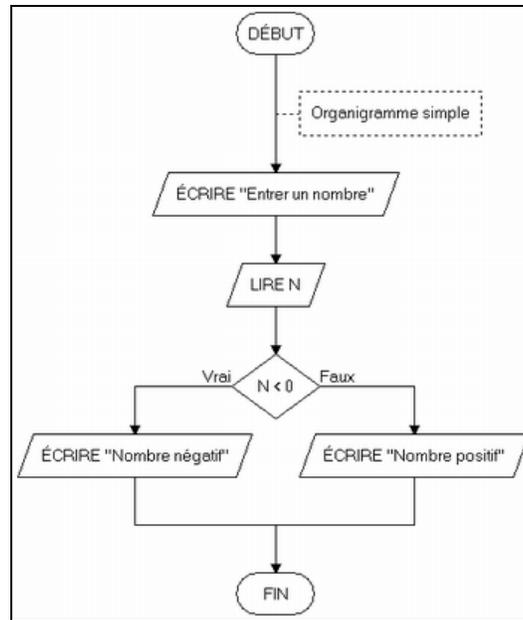
  SI N < 0 ALORS
    ÉCRIRE "Nombre négatif"
  SINON
    ÉCRIRE "Nombre positif"
  FINSI
FIN
```

Pseudo-code 1-1 : Un pseudo-code

Comme vous pouvez le constater, la syntaxe du langage *LARP* est simple et facile à comprendre.

LARP offre aussi la possibilité de formuler des algorithmes sous forme d'organigrammes. Le Pseudo-code 1-1 peut ainsi être formulé sous forme d'organigramme tel qu'illustré par l'Organigramme 1-1. *LARP* offre un [environnement de développement](#) d'algorithmes simple et convivial permettant à un utilisateur novice de se familiariser rapidement avec le logiciel. L'utilisateur peut ainsi consacrer ses énergies à programmer des algorithmes plutôt qu'à se familiariser avec une interface complexe ou une syntaxe de programmation aride.

La flexibilité du langage de programmation *LARP* ainsi que la convivialité de l'environnement de développement rend le logiciel particulièrement propice à l'enseignement de la programmation. L'enseignant peut exploiter le langage *LARP* dans des pseudo-codes et/ou des organigrammes pour présenter de façon claire et concise les concepts de programmation tels les [conditions](#), les [boucles](#) et la [modularité](#). En pratique, les étudiants peuvent exploiter l'environnement de développement de *LARP* pour implémenter et expérimenter les algorithmes présentés par l'enseignant. En fait, *LARP* fut à l'origine développé par un enseignant en informatique dans le but d'enseigner les concepts de la programmation structurée.



Organigramme 1-1 : Un organigramme

Afin de faciliter l'exploitation du langage dans un environnement éducatif, *LARP* est doté d'un [système d'aide contextuel](#) présentant la syntaxe du langage *LARP* sous une forme pédagogique. Ainsi, la documentation en ligne permet à l'utilisateur non seulement d'apprendre la syntaxe de *LARP* afin de programmer des algorithmes, mais aussi d'apprendre à exploiter des notions de programmation telles les [variables](#) et [conteneurs](#), les [structures conditionnelles](#) et [répétitives](#), la [modularité](#) et le [stockage de données](#). Ces notions de programmation sont expliquées et accompagnées d'exemples concrets facilitant leur compréhension.

Le logiciel *LARP* est un outil pédagogique essentiel à l'enseignement de la programmation. Que ce soit en apprentissage autonome ou en classe, *LARP* rend l'apprentissage de la programmation plus facile et agréable.

1.1 Licence d'utilisation

LARP est distribué en version *gratuciel* ainsi qu'en version *partagiciel*. Les deux versions du logiciel sont identiques à l'exception des fonctionnalités suivantes :

- La version *gratuciel* n'offre pas de fonctionnalités de prévention du plagiat alors que la version *partagiciel* l'offre lorsque enregistrée.
- La version *gratuciel* n'offre pas la mise à jour automatisée de l'installation (les mises à jour doivent être récupérées et installées manuellement) alors que la version *partagiciel* l'offre lorsque enregistrée.
- La version *gratuciel* ne bridant pas de fonctionnalités, le mode super-utilisateur n'est pas requis. La version *partagiciel* supporte le mode super-utilisateur lorsque les fonctionnalités de prévention du plagiat sont activées durant l'installation.

Les deux versions du logiciel sont distribuées en un seul fichier d'installation. La version à installer est sélectionnée par l'utilisateur lors du processus d'installation.

1.1.1 Version gratuitiel

LARP est disponible en version *gratuitiel*. Le gratuitiel (en anglais : *freeware*) consiste pour un auteur à mettre en libre circulation une version complète de son logiciel afin de permettre au public de l'utiliser sans frais.

Librement téléchargeable, un gratuitiel est par définition gratuit! Aucun engagement financier ne lie quiconque *installe* le logiciel à son *auteur*. L'utilisateur peut installer et utiliser le gratuitiel sans aucune restriction. Le gratuitiel est cependant soumis à certaines restrictions quant à sa distribution (voir la section 1.1.3 pour les détails).

La version gratuitiel de *LARP* est facilement identifiable via la barre titre de la fenêtre principale de l'*environnement de développement* :



Figure 1-1 : Identification de la version gratuitiel

La *fenêtre d'accueil* du logiciel identifie aussi quelle version de *LARP* est utilisée.

1.1.2 Version partagiciel

LARP est disponible en version *partagiciel*. Le partagiciel (en anglais : *shareware*) consiste pour un auteur à mettre en libre circulation une version complète de son logiciel afin de permettre au public de le tester sans s'engager financièrement. On peut assimiler ce mode de distribution à une période de « libre essai » durant laquelle l'auteur vous offre l'opportunité d'évaluer un logiciel de la façon la plus évidente et la plus efficace qui soit : en l'utilisant comme si vous l'aviez acheté.

Librement téléchargeable, un partagiciel n'est pas pour autant gratuit. Un contrat moral lie en effet son *auteur* à quiconque *installe* le logiciel. Il stipule que l'utilisateur du partagiciel doit, s'il souhaite utiliser le produit et même le conserver sur son ordinateur au terme de la période d'évaluation, s'acquitter du prix de la licence d'utilisation. En clair : une fois écoulée un nombre prédéterminé de jours d'essai du partagiciel, il faudra passer à la caisse si l'on souhaite en garder l'usufruit. Dans le cas contraire, l'utilisateur se devra de reposer le logiciel sur son rayonnage, autrement dit le *désinstaller*, si pour quelque raison que ce soit il ne souhaite pas l'acquérir.

La version partagiciel de *LARP* est facilement identifiable via la barre titre de la fenêtre principale de l'*environnement de développement* :



Figure 1-2 : Identification de la version partagiciel

La *fenêtre d'accueil* du logiciel identifie aussi quelle version de *LARP* est utilisée.

1.1.3 Licence

Voici la licence d'utilisation permettant d'évaluer le logiciel *LARP*. Veuillez lire attentivement ce qui suit avant d'utiliser le logiciel. L'utilisation du logiciel indique que vous acceptez tous les termes de cette licence.

Le logiciel (*LARP*) est distribué en version gratuitiel (gratuite) et partagiciel (non gratuite). L'utilisateur sélectionne la version à installer lors du processus d'installation du logiciel. Les deux versions sont identiques à l'exception des outils de prévention du plagiat qui sont restreints à la version partagiciel.

Les termes de la licence ci-dessous sont applicables aux deux versions du logiciel à l'exception de la première section qui est spécifique à chaque version.

Version gratuitiel

Le logiciel est distribué comme gratuitiel, ce qui signifie qu'il peut être utilisé sans frais ni obligation d'enregistrement. Le gratuitiel (en anglais : *freeware*) consiste pour un auteur à mettre en libre circulation une version complète ou partiellement bridée de son logiciel afin de permettre au public de l'exploiter sans s'engager financièrement.

Voici la licence d'utilisation permettant d'exploiter le gratuitiel. Veuillez lire attentivement ce qui suit avant d'utiliser le gratuitiel. L'utilisation du gratuitiel indique que vous acceptez tous les termes de cette licence.

Version partagiciel

Le logiciel est un partagiciel, ce qui implique qu'il n'est pas gratuit. Ceci signifie que, conformément aux termes énoncés ci-dessous, vous pouvez l'utiliser gratuitement, dans un but d'évaluation, durant une période de 120 jours à compter de son installation. Si vous souhaitez continuer à l'utiliser à l'issue de cette période, vous devrez vous enregistrer auprès de l'auteur (*Marco Lavoie*). Dans le cas contraire, vous devrez supprimer ce logiciel de votre ordinateur.

Voici la licence d'utilisation permettant d'évaluer le partagiciel. Veuillez lire attentivement ce qui suit avant d'utiliser le partagiciel. L'utilisation du partagiciel indique que vous acceptez tous les termes de cette licence.

Distribution

Vous être autorisé à dupliquer le programme d'évaluation (non enregistré) et à le distribuer sans aucune modification à quiconque par des moyens électroniques (Internet, BBS's, CD, etc.).

Vous ne devez demander aucun frais de copie, aucun droit de distribution autre qu'une participation raisonnable relative à vos frais (ex: packaging). Vous ne devez en aucun cas mentionner que vous vendez le logiciel lui-même. Vous ne devez, en aucun cas, demander une quelconque rétribution pour l'utilisation du logiciel lui-même. La distribution de la version d'évaluation ne peut engager aucune compensation d'aucune sorte de la part de l'auteur.

Version enregistrée

Une copie enregistrée de ce logiciel ne peut être utilisée que par une seule et même personne sur un ou plusieurs ordinateurs. Vous pouvez accéder à ce logiciel à travers un réseau à partir du moment où vous avez obtenu autant de licences individuelles que d'ordinateurs susceptibles d'exécuter le programme sur le réseau. Par exemple, si six postes différents peuvent accéder au logiciel, vous avez besoin d'une licence pour six utilisateurs, qu'ils utilisent le logiciel à différents moments ou de manière simultanée.

Restrictions d'utilisation

Vous ne devez pas altérer ce logiciel en aucune façon, ni ajouter, supprimer ou changer un quelconque message ou une quelconque boîte de dialogue.

Vous ne devez pas décompiler, faire du « reverse engineering », désassembler ou transformer ce logiciel en un quelconque code source. Vous ne devez pas modifier, prêter, louer ou vendre ce logiciel.

Vous ne devez pas publier ou distribuer des algorithmes ou utilitaires permettant de générer des codes d'enregistrement, ni publier d'information à propos de ces codes, ni distribuer ou publier ces codes d'enregistrement sans une autorisation écrite de l'auteur.

Restrictions de garantie

Ce logiciel est fourni « tel que » et sans garantie quant à l'exécution, aux performances, à la valeur marchande ou à toutes les autres garanties, qu'elles soient exprimées ou implicites. En raison des divers environnements matériels et logiciels dans lesquels ce programme peut être installé, aucune garantie d'intégration ou de conformité n'est offerte. La bonne pratique informatique dicte que tout nouveau logiciel doit être complètement testé par l'utilisateur avec des données non critiques avant de l'exploiter avec des données critiques. L'utilisateur assume le risque d'utiliser le logiciel. La responsabilité du vendeur est limitée exclusivement au remplacement du produit ou au remboursement du prix d'achat du logiciel.

Copyright

Ce produit est Copyright © 2004-2008 Marco Lavoie.

Il est protégé par la loi canadienne relative aux droits d'auteurs et par les conventions internationales. Vous reconnaissez qu'aucun titre de propriété intellectuelle de ce logiciel ne vous est transféré. Vous reconnaissez en outre que le titre et les pleins droits de ce logiciel demeurent la propriété exclusive de l'auteur, et que vous n'acquerez aucun droit autres que ceux expressément spécifiés dans cet accord de licence.

1.2 Auteur de *LARP*

L'auteur du logiciel *LARP* est :



Nom : Marco Lavoie
M. Sc. Informatique & mathématiques

Adresse postale : 1, Avenue du Parc
Gatineau, Qc (Canada)
J8Y 1G5

Courrier Électronique : info@marcolavoie.ca

Site Web : www.marcolavoie.ca

1.3 Installation

Le programme d'installation du logiciel *LARP* installe sur l'ordinateur le logiciel, ses fichiers connexes et les utilitaires essentiels au fonctionnement de *LARP*. Une fois l'installation terminée et selon les directives d'installation fournies par l'utilisateur, vous pouvez démarrer *LARP* :

- via le dossier **LARP**;
- via le bouton **Démarrer**, sous la rubrique **Programmes** » **LARP**;
- via une icône créée sur le **Bureau** de l'ordinateur;
- via un bouton créé dans la zone de **Lancement rapide** du bureau.

La procédure d'amorçage de l'installation de *LARP* diffère selon que la source du programme d'installation est un **CD** ou un **fichier téléchargé** via Internet.

Le même CD ou fichier téléchargé permet d'installer toutes les versions du logiciel (**gratuciel** ou **partagiciel**) en plusieurs langues dont le français et l'anglais.

1.3.1 Exigences minimales d'équipements et logiciels

Voici la configuration minimale requise pour installer et exploiter *LARP* sur un ordinateur :

Exigences matérielles :

- Ordinateur de type PC
- 64 Mo de mémoire vive
- 5 Mo d'espace libre sur le disque rigide
- Lecteur CD (si *LARP* doit être installé à partir d'un CD)

Exigences logicielles :

- *Microsoft® Windows®* 95, 98, ME, NT, 2000 ou XP.
- *Netscape Navigator* 3.01, *Netscape Communicator* 4.x ou *Microsoft® Internet Explorer* v4.0, ou une version plus récente de ces utilitaires.

1.3.2 Installation à partir d'un CD

Si la source d'installation de *LARP* est un CD, la procédure d'installation est la suivante :

1. Démarrez votre ordinateur et attendez la fin du chargement de *Windows®*.
REMARQUE : Si *Windows®* est déjà lancé, fermez toutes les applications avant de poursuivre l'installation.
2. Si l'ordinateur est doté d'un antivirus, désactivez temporairement ce dernier, le temps d'installer *LARP*.
3. Insérez le CD d'installation de *LARP* (étiquette tournée vers le haut) dans le lecteur CD de l'ordinateur.
4. Si le programme d'installation de *LARP* ne démarre pas automatiquement, vous devez le démarrer manuellement :
 - 4.1. Ouvrez le **Poste de travail** ou lancez l'**Explorateur Windows**.
 - 4.2. Cliquez deux fois sur la lettre correspondant au lecteur CD (en général **D:**, **E:** ou **F:**).
 - 4.3. Cliquez deux fois sur le fichier application **Setup**.

Suivez les instructions à l'écran pour installer le logiciel.

1.3.3 Installation à partir d'un fichier téléchargé

Si le fichier d'installation de *LARP* fut téléchargé via Internet, le nom du fichier téléchargé devrait être **LarpSetup.exe**.

Pour installer *LARP* à partir du fichier téléchargé, la procédure d'installation est la suivante :

1. Démarrez votre ordinateur et attendez la fin du chargement de *Windows®*.
REMARQUE : Si *Windows®* est déjà lancé, fermez toutes les applications avant de poursuivre l'installation.
2. Si l'ordinateur est doté d'un antivirus, désactivez temporairement ce dernier, le temps d'installer *LARP*. Il est cependant recommandé de vous assurer au préalable que le fichier d'installation est exempt de virus.
3. Ouvrez le **Poste de travail** ou lancez l'**Explorateur Windows**.
4. Trouvez le fichier d'installation de *LARP* téléchargé (si vous avez de la difficulté à le localiser sur votre ordinateur, employez la fonction de **Recherche** de l'**Explorateur Windows**).

5. Cliquez deux fois sur le fichier d'installation `LarpSetup.exe`.

Suivez les instructions à l'écran pour procéder à l'installation du logiciel.

1.3.4 Désinstallation

La procédure d'installation de *LARP* installe aussi sur l'ordinateur une procédure automatisée de désinstallation. Celle-ci détruit :

- les fichiers du logiciel *LARP* (les fichiers exécutables, les fichiers de documentation et les exemples de projets *LARP* distribués avec le logiciel), et
- les raccourcis *LARP* localisés sur le **Bureau**, dans la zone de **Lancement rapide** et dans le menu **Démarrer**.

Il est à noter cependant qu'aucun des projets *LARP* créés par les utilisateurs n'est effacé de l'ordinateur par la procédure de désinstallation.

Pour désinstaller complètement le logiciel *LARP*, procédez comme suit :

1. Cliquez sur le bouton **Démarrer**, sélectionnez **Paramètres**, puis **Panneau de configuration**.
2. Cliquez deux fois sur l'icône **Ajout/Suppression de programmes**.
3. Faites défiler la liste et sélectionnez l'item **LARP version #** (où # est le numéro de la version *LARP* installée sur l'ordinateur).
4. Cliquez sur le bouton **Modifier/Supprimer**.

Une fois le logiciel désinstallé, cliquez sur **OK** pour conclure la désinstallation.

1.4 Enregistrement

Cette section du guide réfère exclusivement à la version partagiciel de LARP.

Tel que stipulé dans la [licence d'utilisation](#), vous devez [enregistrer LARP](#) en version partagiciel au terme de la période d'évaluation du logiciel. Dans le cas contraire, vous êtes dans l'obligation de cesser d'exploiter le logiciel et de le [désinstaller](#) de vos ordinateurs.

Lors de la période d'évaluation de *LARP*, une *fenêtre de rappel d'enregistrement* (voir Figure 1-3) est automatiquement affichée au démarrage (et éventuellement à la fermeture) du logiciel. Cette fenêtre énumère les bénéfices de l'enregistrement et indique le nombre de jours restant à la période d'évaluation (la notice clignote lorsque la période d'évaluation est expirée).

Selon la licence d'utilisation à laquelle vous avez accepté d'adhérer lors de l'installation de *LARP* en version partagiciel, vous devez cesser d'évaluer *LARP* aussitôt la période d'évaluation écoulée. Passée la période d'évaluation, vous avez trois alternatives :

1. [Enregistrer](#) votre installation *LARP* afin de continuer à l'utiliser.
2. [Désinstaller LARP](#) de vos ordinateurs et cesser de l'utiliser.
3. Désinstaller *LARP* et installer la [version gratuiciel](#) du logiciel.

Notez que le fait de désinstaller puis réinstaller la version partagiciel de *LARP* afin de contrer l'expiration de la période d'évaluation est prohibée par la licence d'utilisation. Une telle tentative est automatiquement détectée par *LARP*.



Figure 1-3 : Démarrage de *LARP*

1.4.1 Procédure d'enregistrement

Cette section du guide réfère exclusivement à la *version partagiciel* de *LARP*.

Pour enregistrer votre installation de *LARP* en *version partagiciel*, vous devez acheter via Internet une **clé d'enregistrement** (séquence alphanumérique servant à transformer une version d'évaluation du partagiciel en version enregistrée) :

- Si vous êtes disposé à faire votre achat par carte de crédit via site Web sécurisé, consultez le [site Web de LARP](#). La procédure d'enregistrement via Internet est rapide, et vous recevrez votre clé d'enregistrement par courrier électronique dans les minutes suivant votre paiement par carte de crédit.
- Si vous êtes réticent à faire vos achats par carte de crédit via Internet, il est possible de transmettre votre demande d'achat par fax. Le traitement d'une demande d'achat reçue par fax est cependant plus long, donc il s'écoulera quelques jours avant la réception de votre clé d'enregistrement par courrier électronique.
- Si vous n'êtes pas disposé à faire votre achat par carte de crédit, consultez le [site Web de LARP](#) afin d'obtenir plus d'information sur les achats par chèques ou par traites postales.

Lorsque que votre achat est complété et que vous êtes en possession de votre clé d'enregistrement, vous devez accéder à la *fenêtre d'enregistrement* (voir Figure 1-4) via le bouton **Procéder à l'enregistrement** de la fenêtre de rappel d'enregistrement, ou via la barre de menu. Complétez les champs indiqués avec les coordonnées d'enregistrement vous ayant été fournies lors de l'achat (i.e. le nom d'enregistrement et la clé l'accompagnant). Votre version partagiciel de *LARP* sera alors enregistrée et les *rappels d'enregistrement* cesseront.

Si vous devez ultérieurement installer la version partagiciel de LARP sur un autre ordinateur, vous devrez répéter cette procédure d'entrée des coordonnées d'enregistrement. Dans ces circonstances, consultez la [licence d'utilisation](#) régissant l'installation du partagiciel sur plus d'un ordinateur.

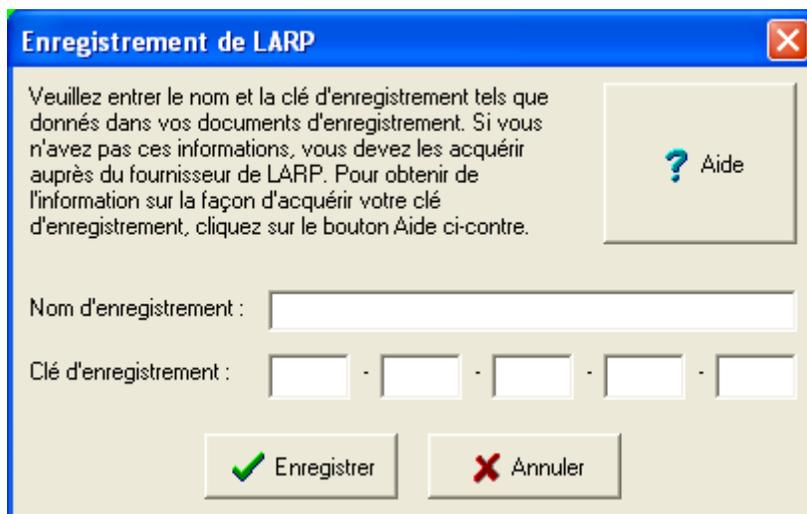


Figure 1-4 : Enregistrement de LARP

1.4.2 Commander des clés de débridage

Cette section du guide réfère exclusivement à la [version partagiciel](#) de LARP.

Une *clé de débridage* est requise pour activer le [mode super-utilisateur](#) de LARP en [version partagiciel](#). Chaque clé est pré-programmée avec un *pseudonyme* unique et non modifiable.



Figure 1-5 : Clé de débridage

Pour commander des clés de débridage, vous devez au préalable posséder une [version enregistrée](#) du partagiciel LARP (lors de la commande de clés, votre nom et clé d'enregistrement ainsi que votre numéro de licence seront exigés). Les coordonnées de votre licence d'utilisation sont affichées dans la *fenêtre d'accueil* de LARP (voir Figure 1-6). Cette fenêtre est accessible en tout temps via la barre de menu.



Figure 1-6 : Fenêtre d'accueil

Les clés de débridage pour *LARP* sont vendues via Internet :

- Si vous êtes disposé à faire votre achat par carte de crédit via site Web sécurisé, consultez le [site Web de LARP](#).
- Si vous êtes réticent à transmettre vos coordonnées de carte de crédit via Internet, il est possible de transmettre votre demande d'achat par fax (consultez le [site Web de LARP](#)).
- Si vous n'êtes pas disposé à faire votre achat par carte de crédit, consultez le [site Web de LARP](#) afin d'obtenir plus d'information sur les achats par chèques ou par traites postales.

Les clés de débridage achetées seront postées par courrier express dès que la transaction financière sera complétée.

1.5 Mises à jour de *LARP*

Cette section du guide réfère exclusivement à la [version partagiciel](#) de *LARP*.

LARP en [version partagiciel](#) dispose d'un système de mise à jour intégré, soit activé explicitement via la [barre de menu](#) ou implicitement à chaque démarrage de l'application. La mise à jour périodique d'une installation *LARP* assure le bon fonctionnement du partagiciel et donne accès aux plus récentes corrections de bogues et aux nouvelles fonctionnalités. Le système de mise à jour intégré est uniquement accessible lorsque le partagiciel *LARP* est [enregistrée](#).

Lorsque le système de mise à jour intégré est activé, *LARP* se branche au serveur Web de distribution des mises à jour afin de vérifier si des nouvelles mises à jour sont disponibles. Si c'est le cas, celles-ci sont automatiquement téléchargées et installées sur l'ordinateur (Figure 1-7), puis *LARP* est redémarré afin de prendre en compte les mises à jour.

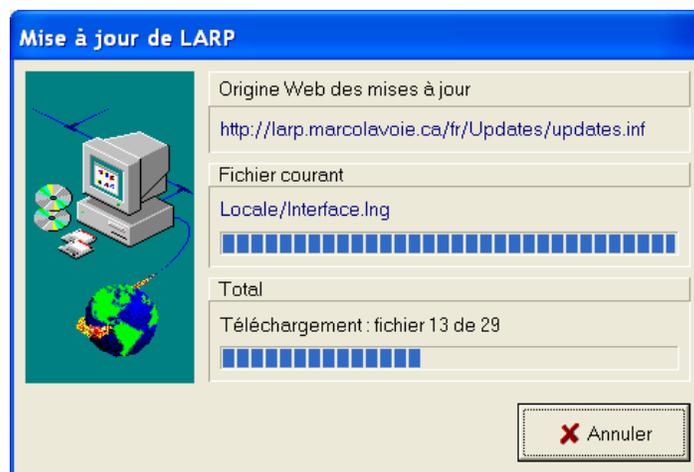


Figure 1-7 : Téléchargement et installation des mises à jour

Le téléchargement des mises à jour peut échouer pour divers raisons, dans quel cas l'installation des mises à jour est annulée et un message d'erreur décrivant l'erreur est affiché. Parmi les causes d'erreur possibles on retrouve :

- **Téléchargement et installation des mises à jour annulés par l'utilisateur** : l'utilisateur a intentionnellement interrompu le processus de mise à jour en appuyant sur le bouton **Annuler**.

- **Connexion Internet interrompue** : le lien à Internet a été interrompu volontairement ou involontairement.
- **Serveur Web détenteur des mises à jour non disponible** : le serveur Web distribuant les mises à jour n'est présentement pas disponible. Dans de telles circonstances il est recommandé de tenter une mise à jour ultérieurement ou contacter le [support technique](#).
- **Répertoire des mises à jour non disponible** : le serveur Web distribuant les mises à jour est défectueux. Contacter le [support technique](#) de *LARP*.
- **Accès aux mises à jour présentement suspendu** : la configuration du serveur Web distribuant les mises à jour est corrompue. Contacter le [support technique](#) de *LARP*.
- **Fichiers de mises à jour manquants sur le serveur Web** : la configuration du serveur Web distribuant les mises à jour est corrompue. Contacter le [support technique](#) de *LARP*.

Il est fortement recommandé d'activer la vérification automatisée de mises à jour au démarrage de *LARP*. Cette option est activée via la fenêtre de [configuration générale](#) du partageiciel. Lorsque l'option est activée et l'ordinateur est relié à Internet, *LARP* se connecte silencieusement au serveur Web de distribution des mises à jour au démarrage, et informe l'utilisateur lorsque de nouvelles mises à jour sont disponibles.

1.6 Support technique

Afin de faciliter l'utilisation de *LARP*, diverses sources d'aide sont mises à la disposition de l'utilisateur :

- *LARP* dispose d'un système d'[aide en ligne](#) offrant une description détaillée de l'[environnement de développement](#) de *LARP* ainsi que son langage pseudo-code et ses éléments d'organigramme.
- De l'[aide abrégée](#) correspondant aux éléments d'interface pointés par la souris est affiché au bas de l'environnement de développement.
- *LARP* dispose d'un système automatisé de [rapports de bogue](#), permettant à l'utilisateur de rapporter tous les bogues rencontrés durant l'utilisation du logiciel.
- Le [site Web de LARP](#) est une excellente source d'information sur *LARP*. On y trouve, entre autres, la plus récente version du logiciel.

Si vous ne trouvez pas réponses à vos questions via ces sources d'information, vous pouvez contacter le support technique de *LARP* via courrier électronique à l'adresse larp@marcolavoie.ca. Notez cependant que le support technique de *LARP* ne s'engage pas à répondre aux questions ou requêtes ayant trait à la logique de conception d'algorithmes.

1.6.1 Aide en ligne

L'aide en ligne de *LARP* est accessible en tout temps en pressant la touche **F1** du clavier ou via la [barre de menu](#), sous la rubrique **Aide**. On y retrouve :

- une description détaillée de l'[environnement de développement](#) de *LARP*,
- une description de la syntaxe du [langage LARP](#) et des instructions d'organigrammes, et
- une description détaillée des [messages d'erreur et d'avertissement](#) affichés par *LARP*.

De multiples exemples de pseudo-codes et d'organigrammes illustrant les caractéristiques du langage *LARP* accompagnent les textes de l'aide en ligne. Diverses notions de programmation y sont aussi présentées sous une forme pédagogique.

Les textes composant l'aide en ligne sont accessibles en format standard *Microsoft® HTML Help*. La plupart des fenêtres affichées dans *LARP* offrent un accès direct aux textes d'aide pertinents via un bouton **Aide** ou via la touche **F1** du clavier.

1.6.2 Rapporter un bogue

Malgré toute l'attention portée par l'auteur de *LARP* à vous offrir un logiciel robuste et sans faille, les erreurs de programmation sont souvent le lot de la plupart des logiciels, y compris probablement *LARP*.

Lorsqu'un bogue survient, soit durant l'édition d'un algorithme ou durant son exécution, dans la plupart des cas un message d'erreur est affiché par *LARP* afin d'en informer l'utilisateur (voir Figure 1-8).

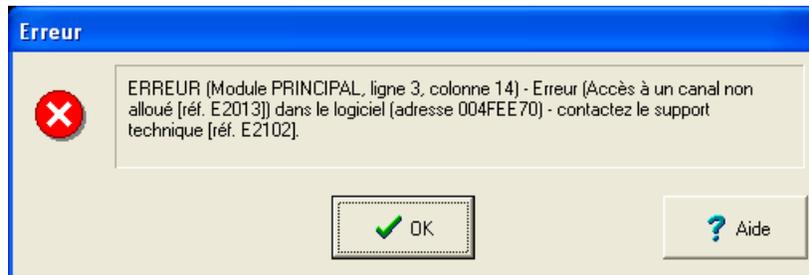


Figure 1-8 : Messages d'erreur

Il peut aussi arriver selon les circonstances que *LARP* affiche un message d'erreur plus détaillé :

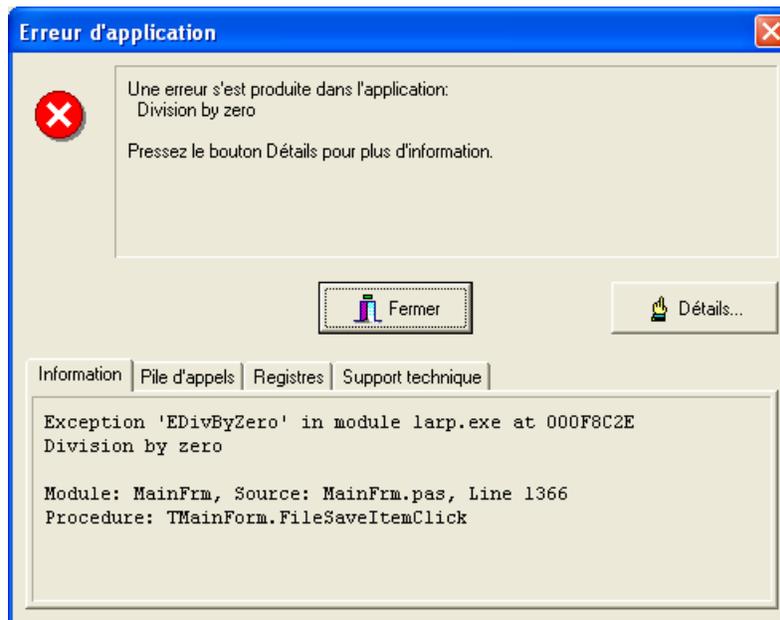


Figure 1-9 : Erreur d'application

La fenêtre intitulée *Erreur d'application* (Figure 1-9), affiche des informations supplémentaires sur la cause du problème ainsi que de l'information facilitant son diagnostic.

En de telles circonstances, il est recommandé d'informer le [support technique](#) du problème afin que le bogue soit éliminé de la prochaine version du logiciel.

LARP dispose d'un système permettant de rapporter les bogues au support technique par courrier électronique. Ce système est accessible via la [barre de menu](#), sous la rubrique **Aide**. Ses caractéristiques sont :

- Un *système de journalisation* enregistre dans un fichier journal tous les [messages d'erreur](#) affichés par LARP.
- Si un projet est chargé dans LARP lors de l'occurrence d'un bogue, une copie du fichier projet est automatiquement sauvegardée pour envoi au support technique de LARP.
- Si l'ordinateur est doté d'un *agent de courrier électronique* (par exemple, *Microsoft® Outlook®*), un courrier électronique peut être automatiquement transmis au support technique de LARP.



Figure 1-10 : Rapport de bogue (formulaire)

Le processus de transmission d'un rapport de bogue via courrier électronique consiste en une séquence de fenêtres servant à colliger de l'information sur les circonstances ayant mené au bogue détecté :

1. La procédure de rapport de bogue débute par une fenêtre permettant à l'utilisateur de s'identifier et de décrire les circonstances ayant mené au bogue (voir Figure 1-10).
2. La seconde fenêtre (Figure 1-11) identifie les fichiers à transmettre au support technique de LARP afin de faciliter l'identification des causes du bogue. L'utilisateur doit autoriser l'envoi de chaque fichier.

3. La troisième fenêtre (Figure 1-12) permet d'initier l'envoi du rapport par courrier électronique via l'agent de courrier électronique de l'ordinateur.

3.1. Si la transmission via l'agent de courrier électronique échoue, un message d'erreur en informe l'utilisateur. Ce dernier peut transmettre le rapport manuellement via un agent de courrier électronique de son choix.

La transmission de l'information colligée permettra à l'équipe du support technique de *LARP* de corriger le bogue afin que la prochaine version de *LARP* en soit exempte.

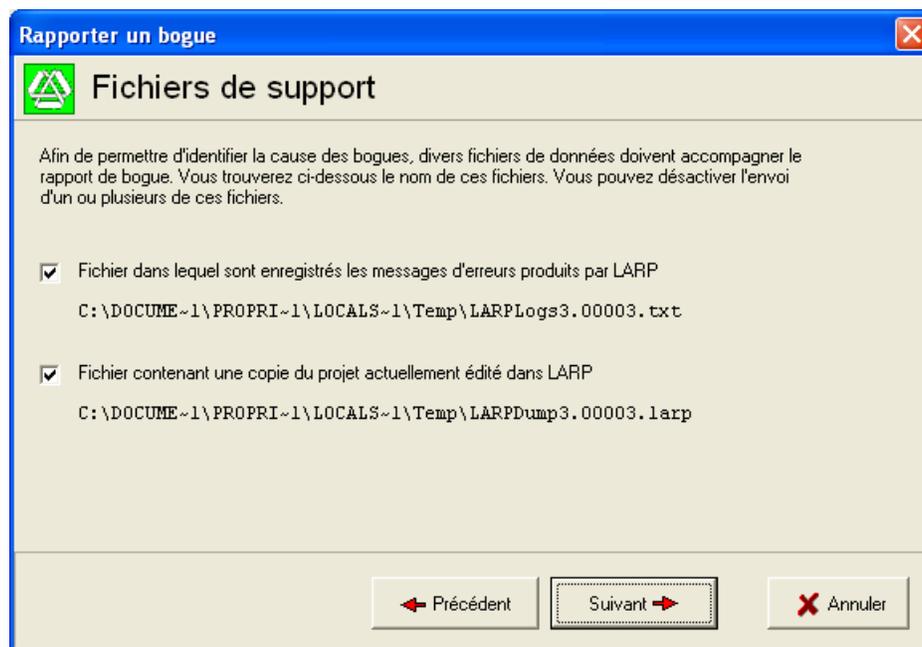


Figure 1-11 : Rapport de bogue (fichiers)

1.6.3 Site Web de *LARP*

L'utilisateur est encouragé à consulter le site Web de *LARP* (larp.marcolavoie.ca) afin d'obtenir de l'aide supplémentaire sur l'utilisation du logiciel. On y retrouve :

- La plus récente version du logiciel.
- Une liste des bogues rapportés et du travail accompli à date pour les corriger.
- Une liste des questions les plus fréquemment posées au support technique de *LARP*, et leurs réponses.
- Des exemples de projets soulignant les diverses fonctionnalités du langage *LARP*.

Si vous ne trouvez pas réponses à vos questions sur le site Web de *LARP*, vous pouvez toujours contacter le support technique via courrier électronique à l'adresse larp@marcolavoie.ca.

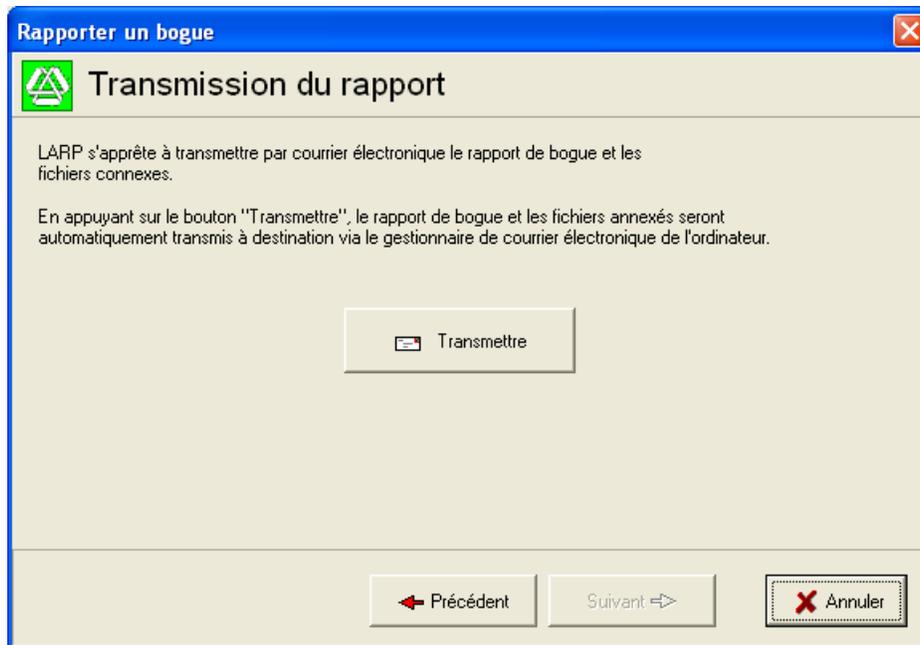


Figure 1-12 : Rapport de bogue (transmission)

2 Environnement de développement

L'environnement de développement de LARP est constitué d'une interface graphique conforme aux standards de *Microsoft® Windows®*. L'utilisateur étant déjà familier avec le type d'environnement accompagnant les outils de développement traditionnels (tels que *Microsoft® Development Studio®* ou *Borland® Delphi®*) n'auront aucune difficulté à maîtriser l'environnement de développement de LARP. Réciproquement, l'utilisateur s'initiant à la programmation avec LARP aura peu de difficultés à transférer les aptitudes acquises à un produit plus traditionnel comme ceux cités ci-haut.

L'environnement de développement de LARP est constitué de plusieurs composants, dont les principaux sont la [fenêtre principale](#), la [console d'exécution](#), la [fenêtre d'exécution pas-à-pas](#) et [l'aide en ligne](#).

2.1 Aide disponible dans LARP

L'aide en ligne de LARP est constituée de fichiers d'aide installés sur l'ordinateur avec le logiciel. Elle peut être invoquée de diverses façons :

- Appuyez sur la touche *F1* en tout temps pour accéder à une page de l'aide en ligne correspondant au contexte d'invocation.
- Invoquez l'aide en ligne via la [barre de menu](#) ou les menus contextuels.
- Plusieurs fenêtres de LARP disposent d'un bouton, nommé **Aide**, permettant d'accéder à l'aide en ligne.
- Lorsqu'un message d'erreur avec numéro de référence est sélectionné dans le [panneau de messages](#), pressez les touches *Ctrl+F1* pour obtenir de l'aide sur l'erreur en question.
- Pressez sur le bouton  du [panneau de contrôle](#) pour obtenir de l'aide contextuel (équivalent à *Ctrl+F1* si un message d'erreur avec référence est sélectionné, *F1* sinon).
- Le [panneau de statut](#) affiche de l'aide abrégée correspondant à l'élément d'interface pointé par la souris.
- Lors de l'installation de LARP, un lien à l'aide en ligne est créé sous la rubrique **Programmes » LARP**, accessible via le menu **Démarrer**.

Le *Guide d'utilisation* de LARP est aussi disponible en version imprimable (format *Acrobat® PDF*). Ce guide contient l'ensemble des pages des textes retrouvées dans l'aide en ligne de LARP.

2.2 Éléments de l'interface

La *fenêtre principale* (Figure 2-1 : Fenêtre principale) de l'environnement de développement de LARP est composée de plusieurs éléments d'interface :

- La [barre de menu](#) donne accès aux commandes de l'environnement de développement (à ne pas confondre avec les instructions pseudo-codes du langage LARP).

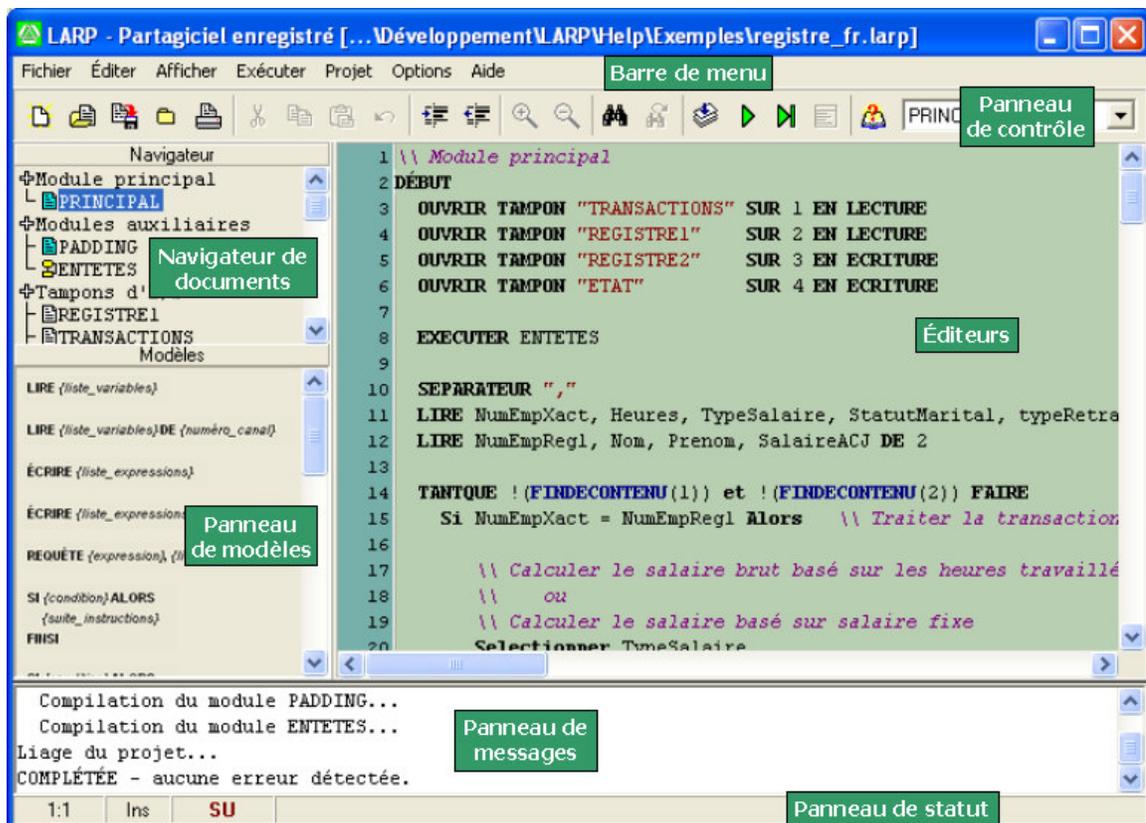


Figure 2-1 : Fenêtre principale

- Le **panneau de contrôle** est constitué d'un ensemble d'éléments d'interface (majoritairement des boutons) donnant un accès rapide à diverses commandes de la barre de menu de *LARP*. Ces boutons correspondent à des commandes qui sont fréquemment invoquées lors de la conception d'algorithmes.
- Les **éditeurs** constituent le panneau central de la fenêtre, où le programmeur transcrit les algorithmes à exécuter. *LARP* dispose de deux éditeurs : l'**éditeur textuel** permet d'éditer les **modules** en pseudo-code ainsi que les **tampons d'entrées/sorties**, et l'**éditeur graphique** permet d'éditer les modules en organigramme.
- Le **panneau de messages** est situé au bas de l'éditeur; *LARP* y affiche divers messages (informations, avertissements et erreurs) habituellement générés lors de la compilation et l'exécution d'algorithmes. Ce panneau peut être optionnellement désactivé via les menus.
- Le **navigateur de documents** énumère les différents modules et tampons d'entrées/sorties contenus dans le projet *LARP* en cours d'édition. L'utilisateur peut visualiser et/ou éditer un document en cliquant sur son nom avec la souris. Ce panneau peut être optionnellement désactivé via les menus.
- Le **panneau de modèles** permet à l'utilisateur d'insérer des composants d'algorithmes lors de l'édition de ceux-ci. Les modèles disponibles dépendent de l'éditeur activé : des modèles d'instructions pseudo-code sont affichés lorsqu'un module en pseudo-code est édité, et des modèles d'instructions d'organigramme sont affichés lorsqu'un module en organigramme est édité.

- Le **panneau de statut**, apparaissant au bas de l'environnement de développement, affiche diverses informations sur l'état courant de LARP. On y retrouve la position du curseur dans l'éditeur, le mode d'insertion actif, le pseudonyme d'utilisateur (uniquement pour LARP en version **partagiciel** avec fonctionnalités de **prévention du plagiat** activées) et l'aide abrégée.

2.2.1 Barre de menu

La *barre de menu* de l'environnement de développement de LARP est située au haut de la fenêtre principale (voir Figure 2-1 : Fenêtre principale). Elle donne accès à toutes les commandes de LARP (à ne pas confondre avec les instructions pseudo-codes du langage LARP).

Le Tableau 2-1 présente la liste des commandes accessibles via la barre de menu. Pour chaque commande sont indiqués la *touche accélératrice* (combinaison de touches du clavier activant la commande), l'*élément d'interface* (É.I.) correspondant à la commande sur le **panneau de contrôle**, et une brève description de la fonctionnalité de la commande. Les commandes identifiées par le symbole § sont des commandes à accès restreint en fonction de la **version de LARP** utilisée et/ou si le **mode super-utilisateur** est activé.

Commande	Accél.	É.I.	Description
Fichier			Regroupe les commandes relatives à l'accès aux fichiers de projets.
Nouveau...	<i>Ctrl+N</i>		Créer un nouveau projet ou un nouveau document dans le projet courant.
Ouvrir...	<i>Ctrl+O</i>		Ouvrir un fichier projet.
Rouvrir	<i>Alt+O</i>		Rouvrir un fichier projet ouvert récemment.
Fermer projet			Fermer le fichier projet courant.
Sauvegarder	<i>Ctrl+S</i>		Sauvegarder le projet courant dans son fichier.
Sauvegarder sous...			Sauvegarder le projet courant dans un fichier spécifié.
Imprimer... (§)	<i>Ctrl+P</i>		Imprimer le projet ou un de ses documents. Cette commande n'est pas accessible dans la version partagiciel de LARP lorsque les fonctionnalités de prévention du plagiat sont activées mais que le mode super-utilisateur ne l'est pas.
Quitter	<i>Alt+F4</i>		Quitter l'application.
Éditer			Regroupe les commandes relatives à l'édition de documents.
Annuler	<i>Ctrl+Z</i>		Annuler la dernière opération d'édition.
Couper	<i>Ctrl+X</i>		Transférer la sélection dans le presse-papiers .
Copier	<i>Ctrl+C</i>		Copier la sélection dans le presse-papiers .

Commande	Accél.	É.I.	Description
Coller	<i>Ctrl+V</i>		Insérer le contenu du presse-papiers au curseur.
Effacer	<i>Del</i>		Efface la sélection.
Effacer tout	<i>Ctrl+Del</i>		Efface la totalité du contenu du tampon d'entrées/sorties .
Sélectionner tout	<i>Ctrl+A</i>		Sélectionner la totalité du contenu du document édité.
Contenu...			Éditer le contenu de l'instruction d'organigramme sélectionné.
Augmenter l'indentation			Augmenter l'indentation (vers la droite) de la ligne au curseur ou du bloc de texte sélectionné dans l' éditeur textuel .
Réduire l'indentation			Réduire l'indentation (vers la gauche) de la ligne au curseur ou du bloc de texte sélectionné dans l' éditeur textuel .
Rechercher...	<i>Ctrl+F</i>		Rechercher des occurrences d'un texte dans les documents du projet.
Rechercher suivant	<i>F3</i>		Répéter la recherche précédente.
Remplacer...	<i>Ctrl+H</i>		Rechercher et remplacer des occurrences d'un texte dans les documents du projet.
Afficher			Regroupe les commandes relatives à l'affichage d'éléments d'interface.
Document			Sélectionner un document du projet afin de l'éditer.
Panneau latéral			Activer / désactiver le navigateur de documents et le panneau de modèles .
Messages			Activer / désactiver le panneau de messages .
Console	<i>F5</i>		Amener à l'avant plan la console d'exécution active.
Agrandir l'affichage			Augmenter le facteur d'agrandissement d'affichage de l' éditeur graphique .
Réduire l'affichage			Réduire le facteur d'agrandissement d'affichage de l' éditeur graphique .
Aucun agrandissement			Annuler le facteur d'agrandissement d'affichage de l' éditeur graphique .
Pseudo-code... (§)			Afficher en pseudo-code l'organigramme dans l' éditeur graphique . Cette commande n'est pas accessible dans la version partagiciel de LARP lorsque les fonctionnalités de prévention du plagiat sont activées mais que le mode super-utilisateur ne l'est pas.

Commande	Accél.	É.I.	Description
Exécuter			Regroupe les commandes relatives à l'exécution de projets.
Compiler...	<i>Ctrl+F7</i>		Compiler le projet (sans l'exécuter).
Exécuter...	<i>F7</i>		Compiler (au besoin) et exécuter le projet.
Exécuter pas-à-pas...	<i>Shift+F7</i>		Compiler (au besoin) et exécuter le projet en mode pas-à-pas .
Interrompre l'exécution...			Terminer immédiatement l' exécution pas-à-pas du projet.
Exécuter un pas	<i>F6</i>		Exécuter la prochaine instruction en mode pas .
Exécution animée	<i>Ctrl+F6</i>		Animer l'exécution de la prochaine instruction en mode pas .
Marcher			Activer l'exécution pas-à-pas en mode marche .
Continuer l'exécution			Activer l'exécution pas-à-pas en mode continu .
Suspendre l'exécution	<i>Shift+F6</i>		Suspendre temporairement l'exécution pas-à-pas du projet.
Point d'arrêt	<i>F8</i>		Activer un nouveau point d'arrêt ou désactiver celui au curseur.
Projet			Regroupe les commandes relatives à la gestion de projets.
Nouveau			Commandes permettant de créer un nouveau projet.
Pseudo-code...			Créer un nouveau projet sous forme de pseudo-code dans le module principal .
Organigramme...			Créer un nouveau projet sous forme d'organigramme dans le module principal .
Fermer			Fermer le projet courant.
Statistiques... (§)	<i>F9</i>		Afficher des statistiques sur la gestion du projet. Cette commande n'est pas accessible dans la version partagiciel de LARP lorsque les fonctionnalités de prévention du plagiat sont activées mais que le mode super-utilisateur ne l'est pas.
Modules			Regroupe les commandes relatives à la gestion des modules du projet courant.
Nouveau			Commandes permettant de créer un nouveau module auxiliaire dans le projet.
Pseudo-code...			Créer un nouveau module auxiliaire exploitant du pseudo-code.
Organigramme...			Créer un nouveau module auxiliaire exploitant de l'organigramme.

Commande	Accél.	É.I.	Description
Renommer...			Modifier le nom du module présentement édité.
Supprimer...			Supprimer du projet le module auxiliaire présentement édité.
Importer... (§)			Importer le contenu d'un fichier texte dans le module en pseudo-code présentement édité. Cette commande n'est pas accessible dans la version partagiciel de <i>LARP</i> lorsque les fonctionnalités de prévention du plagiat sont activées mais que le mode super-utilisateur ne l'est pas.
Exporter... (§)			Exporter le contenu du module en pseudo-code présentement édité dans un fichier texte. Cette commande n'est pas accessible dans la version partagiciel de <i>LARP</i> lorsque les fonctionnalités de prévention du plagiat sont activées mais que le mode super-utilisateur ne l'est pas.
Tampons d'E/S			Regroupe les commandes relatives à la gestion des tampons d'entrées/sorties du projet courant.
Nouveau...			Créer un nouveau tampon d'entrées/sorties dans le projet.
Renommer...			Modifier le nom du tampon d'entrées/sorties présentement édité.
Supprimer...			Supprimer du projet le tampon d'entrées/sorties présentement édité.
Importer...			Importer le contenu d'un fichier texte dans le tampon d'entrées/sorties présentement édité.
Exporter...			Exporter le contenu du tampon d'entrées/sorties présentement édité dans un fichier texte.
Options			Regroupe les commandes relatives à la configuration de <i>LARP</i> .
Générales...			Configurer les options générales de <i>LARP</i> .
Couleurs...			Configurer les options couleurs de <i>LARP</i> .
Enregistrement... (§)			Procéder à l'achat d'une licence d'utilisation . Cette commande n'est pas accessible dans la version gratuitiel de <i>LARP</i> .
Mises à jour... (§)			Télécharger et installer les plus récentes mises à jour du logiciel. Cette commande n'est pas accessible dans la version gratuitiel de <i>LARP</i> .

Commande	Accél.	É.I.	Description
Identification... (§)			Modifier le pseudonyme actif. Cette commande est seulement accessible dans la version partagiciel de LARP lorsque les fonctionnalités de prévention du plagiat sont activées.
Aide			Regroupe les commandes relatives à l'aide en ligne et au support technique.
Aide contextuelle...	F1		Afficher de l' aide en ligne appropriée selon le contexte.
Contenu...	Shift+F1		Afficher la table des matières de l' aide en ligne .
Erreur...	Ctrl+F1		Afficher une description détaillée de l'erreur sélectionnée dans le panneau de messages .
Rapporter un bogue...			Transmettre par courrier électronique un rapport de bogue au support technique de LARP.
À propos...			Afficher la fenêtre d'accueil de LARP.

Tableau 2-1 : Commandes de la barre de menu

2.2.2 Panneau de contrôle

Le *panneau de contrôle* (Tableau 2-2) de LARP regroupe des éléments d'interface (majoritairement des boutons) offrant un accès rapide aux commandes les plus fréquemment invoquées de la *barre de menu* de LARP :



Figure 2-2 : Panneau de contrôle

L'accessibilité aux éléments (boutons et autres) du panneau de contrôle est établie en fonction du contexte (par exemple, le bouton **Couper** n'est activé que lorsqu'un bloc de texte est sélectionné dans l'éditeur textuel ou qu'une instruction d'organigramme est sélectionnée dans l'éditeur graphique).

Voici une description des éléments d'interface (É.I.) apparaissant dans le panneau de contrôle :

É.I.	Commande	Description
	Fichier » Nouveau...	Créer un nouveau projet ou un nouveau document dans le projet courant.
	Fichier » Ouvrir...	Ouvrir un fichier projet.
	Fichier » Fermer projet	Fermer le fichier projet courant.
	Fichier » Sauvegarder	Sauvegarder le projet courant dans son fichier.

É.I.	Commande	Description
	Fichier » Imprimer...	Imprimer le projet ou un de ses documents. Cette commande n'est pas accessible dans la version partagiciel de LARP lorsque les fonctionnalités de prévention du plagiat sont activées mais que le mode super-utilisateur ne l'est pas.
	Éditer » Annuler	Annuler la dernière opération d'édition.
	Éditer » Couper	Transférer la sélection dans le presse-papiers .
	Éditer » Copier	Copier la sélection dans le presse-papiers .
	Éditer » Coller	Insérer le contenu du presse-papiers au curseur.
	Afficher » Augmenter l'indentation	Augmenter l'indentation (vers la droite) de la ligne au curseur ou du bloc de texte sélectionné dans l' éditeur textuel .
	Éditer » Réduire l'indentation	Réduire l'indentation (vers la gauche) de la ligne au curseur ou du bloc de texte sélectionné dans l' éditeur textuel .
	Afficher » Agrandir l'affichage	Augmenter le facteur d'agrandissement d'affichage de l' éditeur graphique .
	Afficher » Réduire l'affichage	Réduire le facteur d'agrandissement d'affichage de l' éditeur graphique .
	Éditer » Rechercher...	Rechercher des occurrences d'un texte dans les documents du projet.
	Éditer » Rechercher suivant	Répéter la recherche précédente.
	Afficher » Console	Amener à l'avant plan la console d'exécution active.
	Exécuter » Compiler...	Compiler le projet (sans l'exécuter).
	Exécuter » Exécuter...	Compiler (au besoin) et exécuter le projet.
	Exécuter » Exécuter pas-à-pas...	Compiler (au besoin) et exécuter le projet en mode pas-à-pas .
	Aide » Aide contextuelle...	Afficher l' aide en ligne . La section affichée dépend du contexte : <ul style="list-style-type: none"> • si une erreur avec référence est sélectionnée dans le panneau de messages, la section de l'aide en ligne décrivant ce type d'erreur est affichée; • si une instruction d'algorithme est sélectionnée dans l'éditeur, la section de l'aide en ligne décrivant cette instruction est affichée; • sinon, la table des matières de l'aide en ligne est affichée.

É.I.	Commande	Description
		Affiche le nom du document édité et permet de changer de document.

Tableau 2-2 : Éléments d'interface du panneau de contrôle

2.2.3 Navigateur de documents

Le *navigateur de documents* (Figure 2-3), situé du côté supérieur gauche de la fenêtre principale de l'*environnement de développement*, affiche les différents *modules* et *tampons d'entrées/sorties* constituant le projet édité :

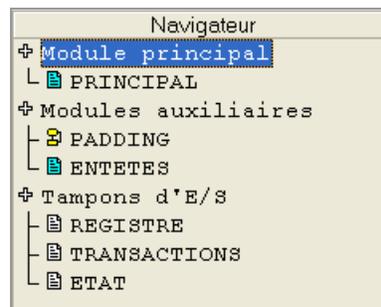


Figure 2-3 : Navigateur de documents

Le nom du document en cours d'édition y est surligné. L'icône accompagnant chaque nom de document indique le type du document : une page bleue (par exemple PRINCIPAL dans la Figure 2-3) pour un module en pseudo-code, un diagramme jaune (PADDING dans la Figure 2-3) pour un module en organigramme et une page blanche (REGISTRE dans la Figure 2-3) pour un tampon d'entrées/sorties. L'utilisateur peut sélectionner un document à éditer en cliquant avec la souris sur le nom du document visé dans le navigateur.

Le navigateur est pourvu d'un menu contextuel (accessible via le bouton droit de la souris) permettant d'ajouter de nouveaux documents au projet, et renommer ou détruire les documents existants.

L'affichage du navigateur de documents est optionnel, pouvant être activé et désactivé via la *barre de menu* de LARP ou via le menu contextuel du navigateur.

2.2.4 Panneau de modèles

Le *panneau de modèles*, situé du côté inférieur gauche de la fenêtre principale de l'*environnement de développement*, offre différents modèles d'instructions d'algorithmes. Les modèles disponibles dans le panneau dépendent du *module* de projet en cours d'édition : lorsque le module est sous forme de pseudo-code, le panneau de modèles affiche des instructions d'algorithmes en pseudo-code (Figure 2-4), alors que lorsque le module est un organigramme, le panneau de modèles affiche des instructions d'organigramme (Figure 2-5). Lorsqu'un *tampon d'entrées/sorties* est édité, le panneau n'offre aucun modèle.

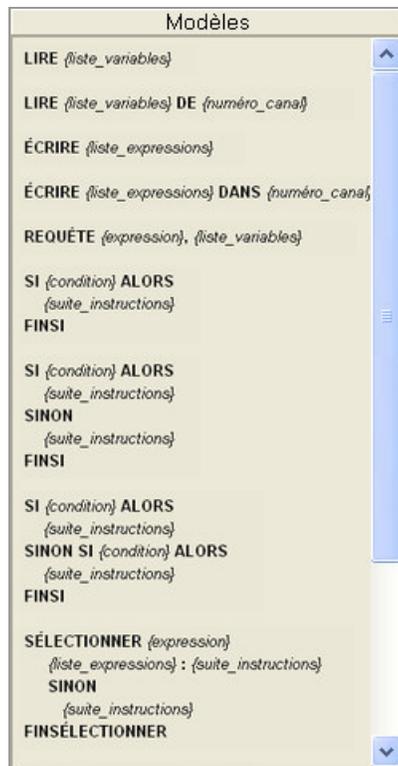


Figure 2-4 : Modèles pour pseudo-code

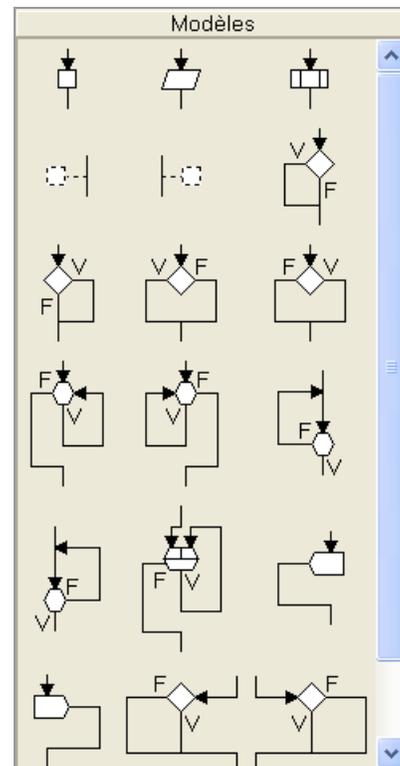


Figure 2-5 : Modèles pour organigramme

Le panneau de modèles permet d'insérer des instructions d'algorithmes dans le module en cours d'édition via des opérations *glisser-déposer* avec la souris. Ces opérations consistent à sélectionner un modèle du panneau avec la souris et à le glisser vers un emplacement approprié dans le module affiché dans l'éditeur. Pour des plus amples informations, consultez les sections [Fonctionnalités de l'éditeur textuel](#) et [Fonctionnalités de l'éditeur graphique](#).

2.2.5 Éditeurs

LARP est doté de deux éditeurs permettant de modifier le contenu des documents d'un projet. L'[éditeur textuel](#) permet d'éditer les modules en pseudo-code et les [tampons d'entrées/sorties](#), alors que l'[éditeur graphique](#) permet d'éditer les modules en organigramme. Ces éditeurs peuvent ouvrir, éditer et sauvegarder les documents énumérés dans le [navigateur de documents](#).

L'éditeur textuel de LARP (Figure 2-6) dispose de fonctionnalités généralement retrouvées dans les éditeurs de texte conventionnels. On y retrouve, entre autres, des fonctions permettant de couper, copier et coller du texte vers et depuis le [presse-papiers](#), [surligner](#) les mots réservés du langage LARP, importer et exporter du texte, afficher le numéro de chaque ligne en marge de la fenêtre d'édition et afficher la position du curseur dans le [panneau de statut](#).

```
1  \\ Module auxiliaire SIGNE
2  \\ Calcule la fonction mathématique SIGNE
3  \\ du paramètre donné
4  ENTRER X
5  SI X < 0 ALORS
6    RES = -1
7  SINON IF X > 0 ALORS
8    RES = 1
9  SINON
10   RES = 0
11  FINSI
12  RETOURNER RES
13
```

Figure 2-6 : Éditeur textuel de LARP

L'éditeur graphique de LARP (Figure 2-7) exploite principalement la technologie glisser-déposer avec le [panneau de modèles](#) pour construire et modifier des organigrammes. Parmi les fonctionnalités offertes on retrouve, entre autres, des fonctions permettant de couper, copier et coller des composants d'organigramme vers et depuis le presse-papiers, insérer, déplacer, réorienter et détruire des composants d'organigramme et [éditer les instructions](#) dans les composants.

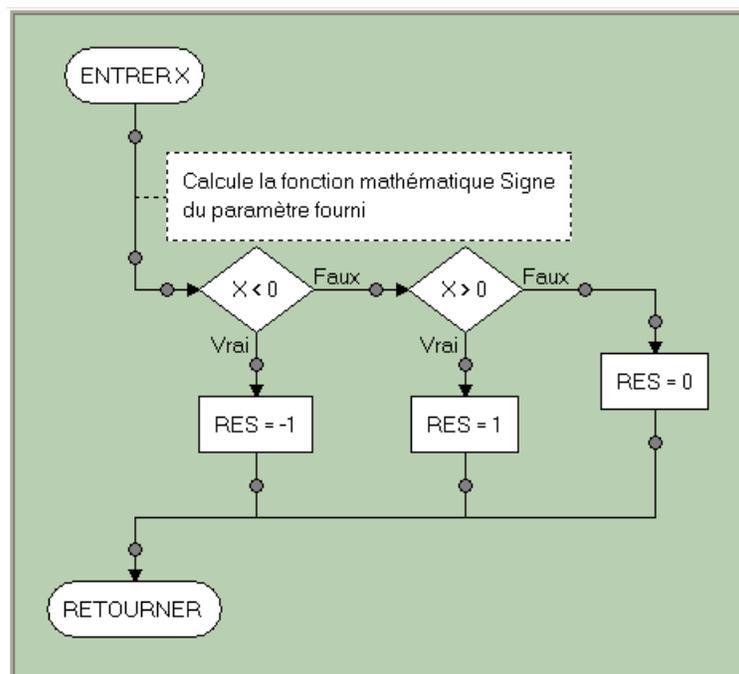


Figure 2-7 : Éditeur graphique de LARP

Les deux éditeurs offrent aussi plusieurs fonctionnalités communes, telles [rechercher et remplacer](#) du texte, annuler les opérations récentes, imprimer le contenu des documents, créer une [sauvegarde de sécurité](#) automatisée des documents et surligner des instructions en [modes d'exécution pas-à-pas](#).

Notez que certaines fonctionnalités des éditeurs sont restreintes selon la [version de LARP](#) utilisée et/ou si le [mode super-utilisateur](#) est activé. Pour plus d'information consultez les sections portant sur les [fonctionnalités de l'éditeur textuel](#) et les [fonctionnalités de l'éditeur graphique](#).

2.2.6 Panneau de messages

Le *panneau de messages* (Figure 2-8), situé sous l'éditeur, est exploité par *LARP* pour afficher divers messages (informations, avertissements et erreurs) généralement générés lors de la [compilation](#) et l'[exécution](#) d'algorithmes :

```
Compilation du projet...
  Compilation du module PRINCIPAL...
  Compilation du module PADDING...
  Compilation du module ENTETES...
Liage du projet...
ERREUR (Module PADDING, ligne 52) - Nombre d'arguments en appel ne correspond pas au
INTERROMPUE - des erreurs ont été détectées.
```

Figure 2-8 : Panneau de messages

Les messages d'information et d'avertissement sont généralement affichés en noir (dépendant des attributs de configuration de *Windows*[®]), alors que les messages d'erreur sont affichés en rouge. Pour localiser l'erreur correspondant à un message, il suffit de cliquer sur le message d'erreur avec la souris et le contenu de l'[éditeur approprié](#) est automatiquement repositionné à l'instruction dans le module où est située l'erreur :

- dans l'[éditeur textuel](#), le curseur est repositionné à l'emplacement dans le pseudo-code où l'erreur est située;
- dans l'[éditeur graphique](#), l'instruction contenant l'erreur est sélectionné.

Le panneau de messages est pourvu d'un menu contextuel (accessible via le bouton droit de la souris) permettant d'effacer les messages affichés et de désactiver le panneau. Celui-ci peut être réactivé via la [barre de menu](#) de *LARP*. Le menu contextuel offre aussi la possibilité d'obtenir plus d'information sur l'erreur détectée (via l'aide en ligne).

Pour plus d'information sur les messages d'avertissement et d'erreur, consultez la section [Compilation et exécution](#).

2.2.7 Panneau de statut

Le *panneau de statut* (Figure 2-9) apparaissant au bas de l'[environnement de développement](#) affiche diverses informations sur l'état courant de *LARP* :

3:5	Ins	57953e8b	Augmenter l'indentation de la ligne au curseur ou du bloc sélectionné.
-----	-----	----------	--

Figure 2-9 : Panneau de statut

On y retrouve les champs d'affichage suivants (de gauche à droite) :

1. le contenu du premier champ dépend de l'éditeur en fonction : y est **affiché la position du curseur** dans l'[éditeur textuel](#) (en format *ligne:colonne*), ou l'**identificateur de l'instruction sélectionnée** dans l'[éditeur graphique](#).

2. le **mode d'insertion actif** ou le **facteur d'agrandissement** : indique si le mode d'insertion de texte est activé (**Ins**) ou non dans l'éditeur textuel, ou indique le facteur d'agrandissement (en %) dans l'éditeur graphique.
3. le **pseudonyme actif** : affiche le **pseudonyme** de l'utilisateur ou **SU** (en rouge) si le mode **super-utilisateur** est activé. Notez que ce champ est affiché uniquement dans la **version partagiciel** de LARP lorsque les fonctionnalités de **prévention du plagiat** sont activées.
4. l'**aide abrégée** : affiche une ligne d'**aide contextuelle** associée à l'élément d'interface sous la souris.

2.2.8 Console d'exécution

Lorsque l'**environnement de développement** de LARP exécute un algorithme, les entrées/sorties sont effectuées via la console d'exécution (Figure 2-10).

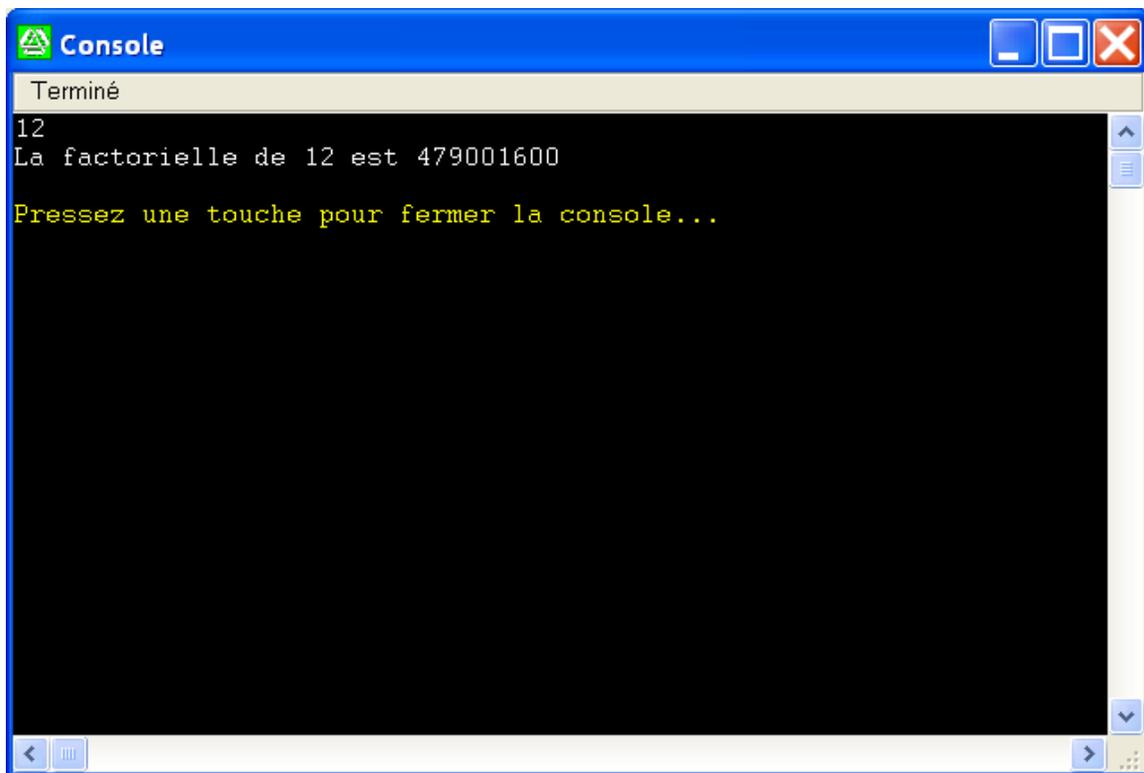


Figure 2-10 : Console d'exécution

La console d'exécution est une fenêtre affichée automatiquement dès le début d'**exécution** de l'algorithme. Cette console permet à l'utilisateur d'interagir avec l'algorithme exécuté, fournissant des valeurs aux instructions de lecture et visualisant les informations affichées par les instructions d'écriture.

Une fois l'exécution de l'algorithme terminée, l'utilisateur doit fermer la console en pressant une touche du clavier. La console peut aussi être fermée en tout temps en fermant la fenêtre (via le bouton de fermeture sur la barre d'en-tête de la fenêtre) ou en interrompant l'**exécution pas-à-pas** de l'algorithme. Si un algorithme est en cours d'exécution lors de la fermeture de la console, une confirmation est demandée avant d'interrompre son exécution (un panneau de statut localisé au haut de la console affiche l'état de l'exécution de l'algorithme : **En exécution...**, **En pause...** ou **Terminé**).

L'environnement de développement de LARP ne peut pas gérer plus d'une console ouverte simultanément. Ainsi, la console ouverte doit être fermée avant que l'algorithme puisse être exécuté de nouveau.

Les barres de déroulement à droite et au bas de la console permettent de visualiser un contenu affiché plus grand que la capacité visuelle de la fenêtre (la fenêtre peut afficher 25 lignes de 80 caractères, mais la console retient jusqu'aux 200 dernières lignes d'entrées/sorties). La fenêtre de la console peut être redimensionnée en tout temps avec la souris.

Alors que le fond de console est toujours de couleur noire, la couleur du texte y étant affiché peut être configurée (voir la section sur la [sélection des couleurs](#)).

2.2.9 Fenêtre d'exécution pas-à-pas

Lorsque l'exécution pas-à-pas de l'algorithme est activée, la *fenêtre d'exécution pas-à-pas* (Figure 2-11) est affichée en avant plan de la *console d'exécution* :

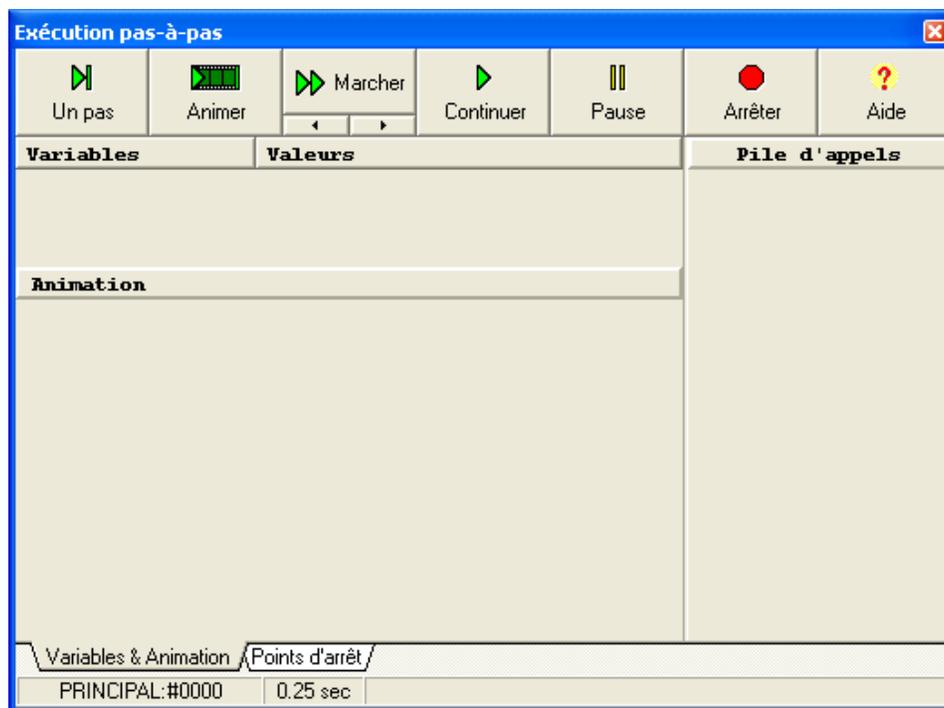


Figure 2-11 : Fenêtre d'exécution pas-à-pas

La fenêtre d'exécution pas-à-pas permet de contrôler l'exécution pas-à-pas de l'algorithme en offrant les fonctionnalités suivantes :

- **Exécution unitaire des instructions** : les instructions de l'algorithme peuvent être exécutées individuellement de façon séquentielle, avec une pause entre chaque instruction afin de permettre l'inspection des variables.
- **Inspection des variables** : entre l'exécution de chaque ligne d'instruction de l'algorithme, la valeur stockée dans chaque variable peut être visualisée. Les éléments de [conteneurs](#) peuvent aussi être inspectés individuellement.

- **Inspection de la pile d'appels** : la pile d'appels consiste en la séquence d'appels de modules exécutés depuis le **module principal** jusqu'au module en cours d'exécution. La fenêtre d'exécution pas-à-pas affiche en continu l'état de la pile d'appels correspondant à l'exécution en cours.
- **Gestion des points d'arrêt** : un point d'arrêt indique une ligne d'algorithme à laquelle l'exécution pas-à-pas de l'algorithme doit être momentanément suspendu. La fenêtre d'exécution pas-à-pas permet de gérer les points d'arrêts actifs.
- **Animation des instructions** : la fenêtre d'exécution pas-à-pas peut animer l'exécution d'une instruction d'algorithme. L'animation permet de visualiser l'ordre d'évaluation des composants de l'instruction.

L'exécution pas-à-pas est décrite en détails dans la section 2.5.2, intitulée Exécution pas-à-pas.

2.3 Fonctionnalités de l'éditeur textuel

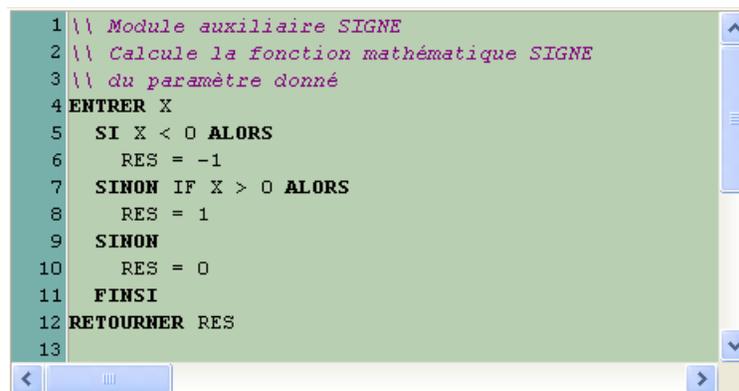
L'environnement de développement de LARP est doté d'un éditeur de texte permettant de modifier le contenu des documents textuels intégrés à un projet (i.e. les **modules** en pseudo-code et les **tampons d'entrées/sorties**). L'éditeur peut ouvrir, éditer et sauvegarder ces documents du projet (énumérés dans le **navigateur de documents**). Il est aussi doté de fonctionnalités généralement retrouvées dans la plupart des éditeurs de texte conventionnels, telles que le copier-coller, la surligne des mots réservés et l'indentation automatisée.

2.3.1 Éditer un document textuel

Pour éditer un document textuel du projet chargé dans LARP (**module** de pseudo-code ou **tampon d'entrées/sorties**), il suffit de sélectionner le document visé dans le **navigateur de documents** ou via le **panneau de contrôle**. Le contenu du document est alors affiché dans l'éditeur. En fait, l'éditeur textuel de LARP est automatiquement affiché lorsque le document édité est un module en pseudo-code ou un tampon d'entrées/sorties.

Le panneau de l'éditeur est composé de deux sections :

- la marge latérale affiche les numéros de lignes, les **signets** et divers identificateurs d'**exécution pas-à-pas**;
- la section principale, à droite, affiche le contenu du document sélectionné pour édition.



```
1  \\ Module auxiliaire SIGNE
2  \\ Calcule la fonction mathématique SIGNE
3  \\ du paramètre donné
4  ENTRER X
5  SI X < 0 ALORS
6    RES = -1
7  SINON IF X > 0 ALORS
8    RES = 1
9  SINON
10   RES = 0
11  FINSI
12  RETOURNER RES
13
```

Figure 2-12 : Panneau de l'éditeur textuel

La couleur de fond du panneau d'édition (la section de droite) est dépendante du type de document édité : un fond vert (couleur par défaut, modifiable via la [sélection des couleurs](#)) indique que le document est un module (contenant du pseudo-code), alors qu'un fond blanc (couleur par défaut, aussi modifiable via la configuration des couleurs) indique que le document est un tampon d'entrées/sorties contenant des données. Lorsque le document édité est un module, la [surligne des mots réservés](#) du langage LARP est automatiquement activée.

Les commandes d'édition de documents sont accessibles via les menus (la [barre de menu](#) ou le menu contextuel accessible en cliquant sur le panneau d'édition avec le bouton droit de la souris), via des boutons sur le [panneau de contrôle](#), [via le clavier](#) et/ou [via la souris](#). L'accès à certaines de ces commandes peut être bridé (i.e. limité au [mode super-utilisateur](#)) dans la [version partagiciel](#) de LARP, comme par exemple la [gestion du presse-papiers](#) et l'[impression de documents](#).

2.3.2 Recherche et remplacement

L'éditeur textuel est pourvu de fonctionnalités de recherche et remplacement de texte. La *fenêtre de recherche* (Figure 2-13) permet de rechercher une séquence de caractères dans le document en cours d'édition ou dans l'ensemble du projet. La fenêtre de recherche et remplacement (Figure 2-14) offre en plus la possibilité de remplacer les occurrences trouvées par une séquence de caractères alternative.

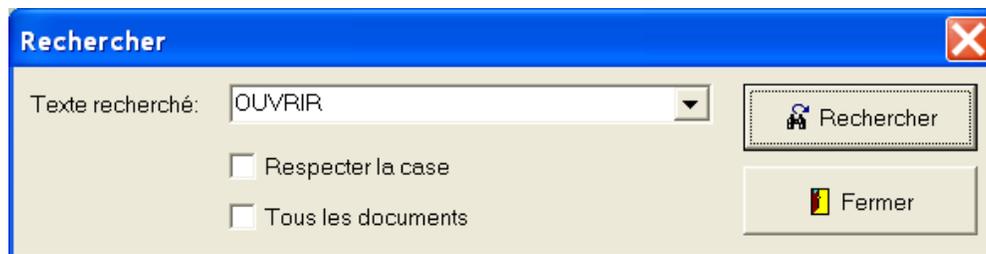


Figure 2-13 : Fenêtre de recherche

Dans la fenêtre de recherche comme dans celle de recherche et remplacement, les séquences de caractères spécifiées antérieurement sont récupérables via les listes de déroulement.

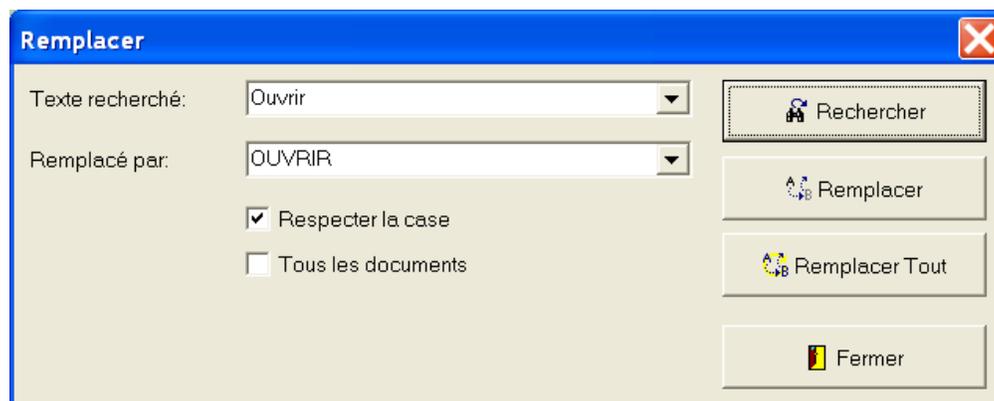


Figure 2-14 : Fenêtre de recherche et remplacement

Les fonctionnalités de recherche et remplacement s'appliquent aussi aux [organigrammes](#).

2.3.3 Surligne de contenu

Lorsque le pseudo-code d'un module est affiché dans l'éditeur textuel, les instructions du langage *LARP* sont affichées de couleurs distinctes selon leur signification. Par défaut, ces couleurs sont :

- Les mots réservés du langage *LARP* sont affichés en **caractères gras noirs**.
- Les fonctions prédéfinies du langage *LARP* sont affichées en **caractères gras bleus**.
- Les commentaires sont affichés en *caractères italiques magentas*.
- Les chaînes de caractères sont affichées en "caractères rouges".

Les autres composants de pseudo-code sont affichés en caractères noirs.

Lors de l'**exécution pas-à-pas** du module, les lignes de pseudo-code suivantes sont surlignées :

- La prochaine instruction à être exécutée en **mode d'exécution par pas**.
- Les instructions étant associées à un **point d'arrêt**.

L'attribution des couleurs de surligne est configurable via la **sélection des couleurs**.

2.3.4 Configuration de l'éditeur textuel

Les caractéristiques suivantes de l'éditeur textuel sont configurables :

- les **fonctionnalités d'édition**,
- les **couleurs d'affichage et de surligne** de pseudo-code.

Pour plus d'information sur la configuration de l'éditeur, consultez les sections correspondantes (pages 72 et 77).

2.3.5 Commandes d'édition de l'éditeur textuel

Les commandes d'édition ainsi que celles de gestion de projet sont accessibles via :

- la **barre de menu**,
- les **menus contextuels**,
- le **panneau de contrôle**,
- le **clavier**, et
- la **souris**.

2.3.5.1 Commandes de l'éditeur textuel accessibles via les menus

On retrouve dans *LARP* deux types de menus :

1. la **barre de menu**, affichée au haut de la fenêtre principale de l'**environnement de développement**, et

2. les menus contextuels, associés à divers éléments d'interface de l'environnement de développement et accessibles via le bouton droit de la souris lorsque celle-ci est positionnée sur l'élément d'interface visé.

L'éditeur textuel répond à diverses commandes de la barre de menu (voir la section portant sur la [barre de menu](#)) ainsi qu'à celles dans son menu contextuel. L'accessibilité à ces commandes dépend du type de document édité :

- Le menu contextuel associé à l'éditeur lors de l'édition d'un [module](#) en pseudo-code affiche les commandes de gestion du [presse-papiers](#) et de [recherche et remplacement](#).
- Le menu contextuel associé à l'éditeur lors de l'édition d'un [tampon d'entrées/sorties](#) (contenant des données) affiche, en plus des commandes de gestion du presse-papiers et de recherche et remplacement, les commandes d'importation et d'exportation de données.

Pour une description détaillée des commandes accessibles via les menus, consultez les sections décrivant la [barre de menu](#) et le [panneau de contrôle](#).

2.3.5.2 Commandes de l'éditeur textuel accessibles via le clavier

En plus des [commandes accessibles via les menus](#), l'éditeur textuel de *LARP* répond à des commandes exclusivement accessibles via une touche ou une combinaison de touches du clavier :

Touches	Description
<End>	Déplace le curseur à la fin de la ligne au curseur.
<Home>	Déplace le curseur au début de la ligne au curseur.
<Enter>	Insère une nouvelle ligne.
<Ins>	Active et désactive le mode d'insertion.
	Supprime le caractère à droite du curseur.
<Backspace>	Supprime le caractère à gauche du curseur.
<Tab>	Insère une tabulation (contextuelle ou conventionnelle) à droite du curseur.
<Left>	Déplace le curseur au caractère précédent.
<Right>	Déplace le curseur au caractère suivant.
<Up>	Déplace le curseur à la ligne précédente.
<Down>	Déplace le curseur à la ligne suivante.
<Page Up>	Déplace le curseur vers le haut d'un nombre de lignes correspondant à la hauteur du panneau d'édition.
<Page Down>	Déplace le curseur vers le bas d'un nombre de lignes correspondant à la hauteur du panneau d'édition.
<Ctrl>+<Up>	Déplace le curseur au début du document.
<Ctrl>+<End>	Déplace le curseur à la fin du document.
<Shift>+<Left> <Right> <Up> <Down>	Sélectionne un bloc de texte conventionnel.
<Alt>+<Left> <Right> <Up> <Down>	Sélectionne un bloc de texte vertical.

Touches	Description
<Ctrl>+<Shift>+0...9	Active ou désactive un signet (les signets étant numérotés de 0 à 9) .
<Ctrl>+0...9	Déplace le curseur au signet correspondant.

Tableau 2-3 : Commandes de l'éditeur textuel accessibles via le clavier

En plus des commandes ci-dessus, *LARP* répond aussi aux touches accélératrices associées à la [barre de menu](#).

2.3.5.3 Contrôle de l'éditeur textuel via la souris

L'éditeur textuel de l'[environnement de développement](#) répond de façon standard aux interactions suivantes via la souris :

- Un clic du bouton gauche de la souris déplace le curseur à la position pointée par la souris dans le document édité.
- Un clic du bouton droit de la souris affiche le menu contextuel à la position pointée par la souris.
- Un clic avec glissement du bouton gauche de la souris sur du texte non-sélectionné de l'éditeur permet de sélectionner un bloc de texte.
- Un clic avec glissement du bouton gauche de la souris sur un bloc de texte sélectionné de l'éditeur permet de déplacer ce bloc de texte à la nouvelle position du curseur.

De plus, la souris peut être employée conjointement avec le [panneau de modèles](#) pour insérer des instructions dans un pseudo-code via des opérations *glisser-déposer*. Pour ce faire il faut sélectionner un modèle dans panneau de modèles, le glisser à la ligne où l'instruction doit être insérée dans l'éditeur textuel, et relâcher le bouton de la souris afin d'y déposer l'instruction correspondant au modèle sélectionné. L'instruction peut ensuite être modifiée afin de l'adapter aux besoins de l'algorithme.

2.4 Fonctionnalités de l'éditeur graphique

L'[environnement de développement](#) de *LARP* est doté d'un éditeur graphique permettant de modifier le contenu des [modules](#) en organigramme. L'éditeur peut ouvrir, éditer et sauvegarder ces documents du projet (énumérés dans le [navigateur de documents](#)). L'utilisateur exploite principalement la technologie glisser-déposer à l'aide du [panneau de modèles](#) pour de construire et modifier des organigrammes. Parmi les fonctionnalités offertes on retrouve, entre autres, celles permettant de couper, copier et coller des instructions d'organigramme vers et depuis le presse-papiers, ainsi que celles permettant d'insérer, déplacer, réorienter, transformer et détruire des instructions d'organigramme et [éditer le contenu](#) de ces instructions.

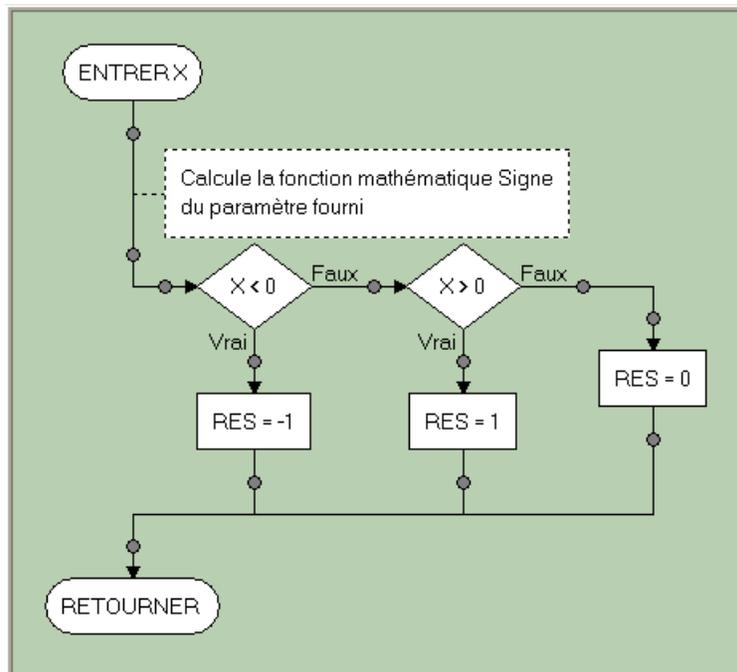
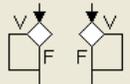
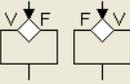


Figure 2-15 : Panneau de l'éditeur graphique

2.4.1 Instructions d'organigramme

Le [panneau de modèles](#) énumère toutes les instructions d'organigramme disponibles dans LARP :

Instructions	Descriptions
	<u>Instruction séquentielle</u> : permet de formuler des instructions séquentielles, telles que l' affectation , l'ouverture et la fermeture de canaux d'entrées/sorties , etc.
	<u>Instruction de lecture et/ou d'écriture</u> : permet de formuler des instructions d' entrées et de sorties d'information .
	<u>Invocation d'un module auxiliaire</u> : appel à un module auxiliaire (pseudo-code ou organigramme) du projet.
	<u>Commentaire</u> : insère des informations non exécutables dans l'organigramme.
	<u>Structure conditionnelle SI</u> : une séquence d'instructions exécutée uniquement en fonction du résultat de l'évaluation d'une condition .
	<u>Structure conditionnelle SI-SINON</u> : deux séquences d'instructions dont une et une seule est exécutée en fonction du résultat de l'évaluation d'une condition .

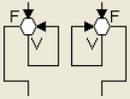
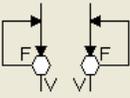
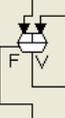
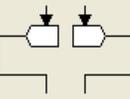
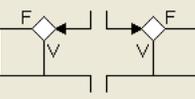
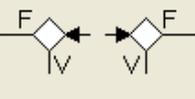
Instructions	Descriptions
	<p>Structure répétitive TANTQUE : une séquence d'instructions exécutée à répétition en fonction du résultat de l'évaluation d'une condition.</p>
	<p>Structure répétitive RÉPÉTER-JUSQU'À : une séquence d'instructions exécutée à répétition en fonction du résultat de l'évaluation d'une condition.</p>
	<p>Structure répétitive POUR : une séquence d'instructions exécutée à répétition un nombre prédéfini de fois.</p>
	<p>Structure de sélection : structure conditionnelle comportant plusieurs séquences d'instructions alternatives dont une seule est exécutée en fonction de la valeur d'une expression arithmétique.</p>
	<p>Structure conditionnelle SI-SINON-SI : structure conditionnelle comportant plusieurs séquences d'instructions alternatives dont une seule est exécutée en fonction du résultat de l'évaluation de conditions.</p>
	<p>Branchement pour structures conditionnelles : permettent d'insérer des branchements conditionnels supplémentaires dans les structures de sélection et les structures conditionnelles SI-SINON-SI.</p>

Tableau 2-4 : Instructions d'organigramme

Notez que plusieurs instructions offrent deux alternatives d'orientation (par exemples les structures conditionnelles). L'orientation d'une telle instruction est purement esthétique et n'a aucun impact sur l'exécution de l'organigramme.

2.4.2 Éditer un organigramme

Pour éditer un document du projet chargé dans LARP (module ou tampon d'entrées/sorties), il suffit de sélectionner le document visé dans le navigateur de documents ou via le panneau de contrôle. Le contenu du document est alors affiché dans l'éditeur approprié. L'éditeur graphique de LARP est automatiquement affiché lorsque le document édité est un module en organigramme. Lorsque le document est composé de texte (module en pseudo-code ou tampon d'entrées/sorties), c'est l'éditeur textuel qui est affiché.

Lorsqu'un nouveau module en organigramme est créé (via la barre de menu ou via le navigateur de documents), un organigramme minimal est automatiquement généré par LARP :

- Si c'est un module principal de projet, l'organigramme comprend les instructions **DÉBUT** et **FIN**.
- Si c'est un module auxiliaire, l'organigramme comprend les instructions **ENTRER** et **RETOURNER**.

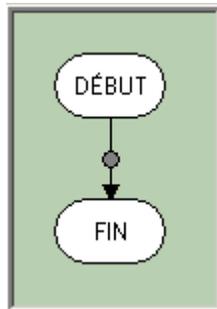


Figure 2-16 : Nouveau module principal

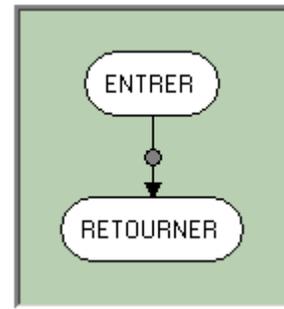


Figure 2-17 : Nouveau module auxiliaire

L'éditeur graphique affiche un *nœud d'insertion* entre chaque instruction d'organigramme. Ce nœud permet d'insérer des instructions dans l'organigramme via son [menu contextuel](#) ou via des [opérations glisser-déposer](#) à partir du [panneau de modèles](#).

2.4.3 Manipulation d'instructions d'organigramme

Le panneau de l'éditeur graphique de *LARP* consiste en un canevas où les instructions de l'organigramme en cours d'édition peuvent être manipulées (voir Figure 2-15). Sur chaque lien reliant deux instructions d'organigramme apparaît un *nœud d'insertion* (petit cercle gris) permettant d'insérer de nouvelles instructions.

Les instructions et les nœuds d'insertion d'un organigramme peuvent être sélectionnés en cliquant sur le composant graphique avec la souris ou via les [touches directionnelles du clavier](#) (entre autres les flèches). La sélection est indiquée visuellement par un rectangle pointillé encadrant l'instruction ou le nœud (voir Figure 2-18 et Figure 2-19).

Une instruction sélectionnée peut être déplacée vers un nœud d'insertion, modifiée ou détruite. Par contre, les nœuds d'insertions ne servant qu'à recevoir des instructions par [glisser-déposer](#) ou via les [menus](#), ils ne peuvent être déplacés ni détruits.

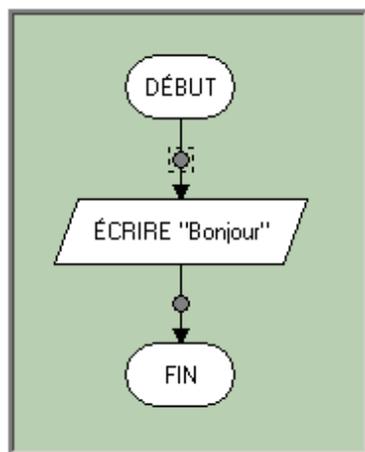


Figure 2-18 : Nœud d'insertion sélectionné

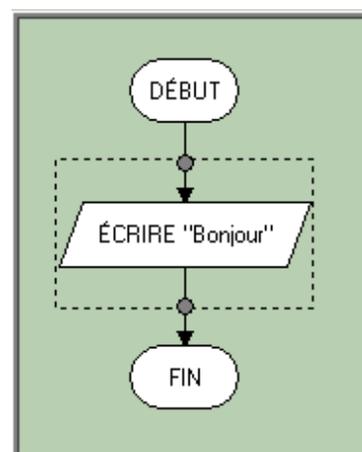


Figure 2-19 : Instruction d'organigramme sélectionnée

2.4.3.1 Insertion, déplacement et destruction d'instructions d'organigramme

Des nouvelles instructions peuvent être insérées dans l'organigramme édité en exploitant les *nœuds d'insertion* :

- via le menu contextuel affiché lorsqu'un nœud est sélectionné à l'aide du bouton droit de la souris ou via les [touches du clavier](#) (Figure 2-20), ou
- via glisser-déposer, en glissant un modèle d'instruction du [panneau de modèles](#) vers un nœud d'insertion et en l'y déposant.

Lorsqu'un modèle d'instruction est glissé vers un nœud d'insertion de l'organigramme, la couleur résultante du nœud visé indique si le modèle glissé peut y être déposé : un nœud **rouge** indique qu'il est logiquement erroné d'y déposer le modèle glissé, un nœud **vert** autorisant le dépôt du modèle.

Une instruction d'organigramme existante peut être déplacée vers un autre nœud d'insertion de l'organigramme par glisser-déposer. Voici les étapes permettant de réaliser un tel déplacement d'instruction :

1. Sélectionner l'instruction de l'organigramme à déplacer en cliquant sur celle-ci avec la souris;
2. glisser l'instruction sélectionnée vers le nœud d'insertion correspondant au nouvel emplacement de l'instruction dans l'organigramme;
3. déposer l'instruction glissée sur le nœud visé en relâchant le bouton de la souris.

La couleur affichée par le nœud destinataire lors du glissement indique si l'instruction peut y être déposée : un nœud **rouge** indique qu'il est logiquement erroné d'y déposer l'instruction, alors qu'un nœud **vert** autorise le dépôt de l'instruction. Si l'instruction glissée n'est pas déposée sur un nœud acceptant celle-ci, le déplacement est annulé.

Les déplacements d'instructions dans l'organigramme peuvent aussi être effectuée par des opérations couper-coller, via le [clavier](#) ou les [menus contextuels](#). Notez cependant que la gestion du presse-papiers est soumise aux [restrictions du mode super-utilisateur](#).

L'instruction sélectionnée peut être éliminée de l'organigramme via la [barre de menu](#), via le [menu contextuel](#) de l'instruction ou en appuyant sur la touche d'effacement (**Del**) du [clavier](#).

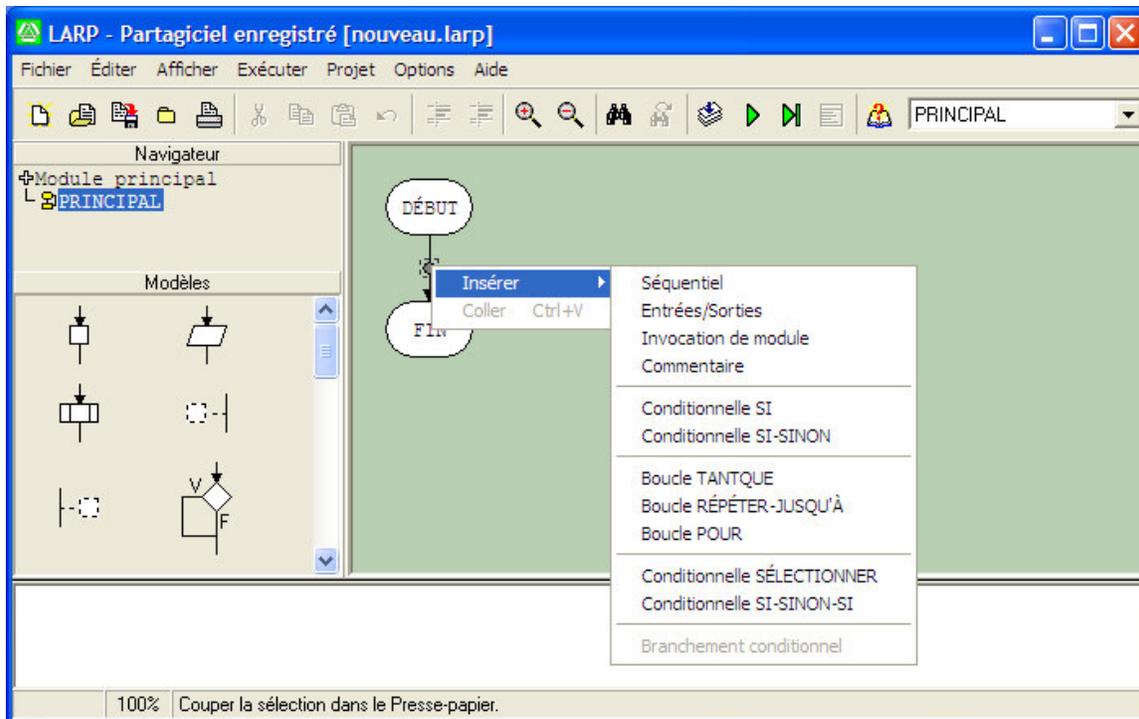


Figure 2-20 : Insertion d'une instruction d'organigramme via le menu contextuel du noeud

2.4.3.2 Édition d'instructions d'organigramme

Les attributs de l'instruction d'organigramme sélectionnée dans l'éditeur graphique peuvent être édités en double cliquant sur celle-ci ou via le menu contextuel associé à l'instruction sélectionnée (accessible en cliquant sur la sélection avec le bouton droit de la souris). La *fenêtre d'édition d'instruction d'organigramme* (Figure 2-21) affiche les attributs associés à l'instruction sélectionnée et permet de modifier ceux-ci.

Les attributs généralement associés à une instruction d'organigramme dépendent de l'instruction visée. Certaines instructions sont relativement simples (ex: les *opérations séquentielles* et les *commentaires*), alors que d'autres sont plus complexes (*structures conditionnelles* et *structures répétitives*), pouvant même inclure un attribut d'orientation telle la structure conditionnelle *SI-SINON* de la Figure 2-21.

Les menus de *LARP* permettent aussi diverses opérations d'édition sur l'instruction d'organigramme sélectionnée, incluant :

- le *changement d'orientation*, consistant à intervertir la direction des branchements dans une structure conditionnelle ou une structure répétitive, et/ou
- la *transformation* de l'instruction en une autre instruction de la même famille (par exemple convertir un *boucle TANQUE* en une *boucle RÉPÉTER-JUSQU'À*).

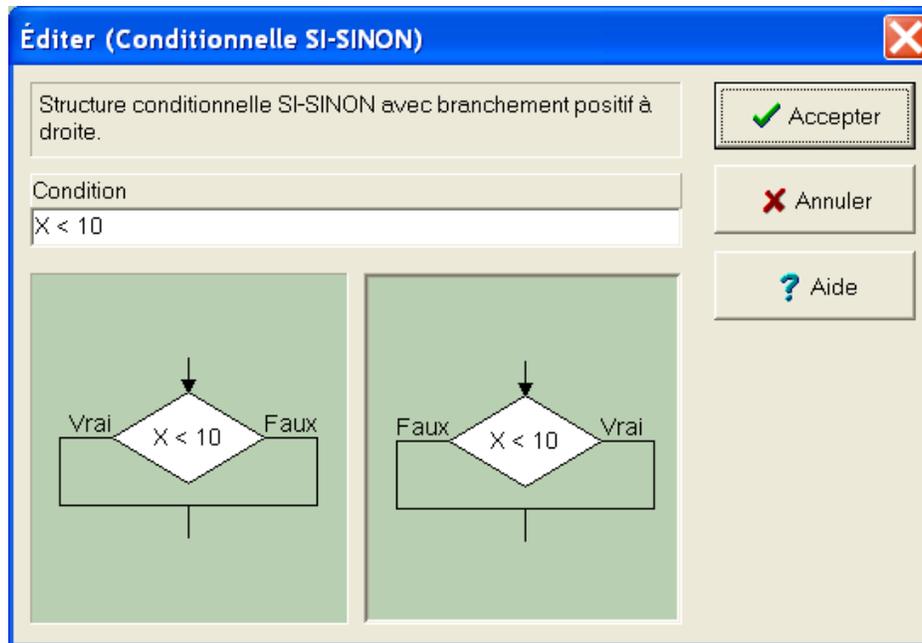


Figure 2-21 : Édition d'une instruction d'organigramme

Les procédures d'édition d'instructions d'organigramme (avec la souris ou via les menus) sont relativement intuitives et vites maîtrisées après quelques minutes d'utilisation de l'éditeur graphique de LARP.

2.4.4 Recherche et remplacement dans un organigramme

Les opérations de recherche et remplacement de LARP sont accessibles dans l'éditeur textuel ainsi que dans l'éditeur graphique.

Toutes les occurrences du texte recherché dans une instruction d'organigramme sont surlignées par l'éditeur graphique. Similairement, une opération de remplacement s'applique simultanément à toutes les occurrences du texte recherché dans l'instruction.

Pour plus d'information sur les fonctions de recherche et remplacement dans LARP, consultez la [section correspondante pour l'éditeur textuel](#).

2.4.5 Agrandissement de l'affichage

L'éditeur graphique permet d'ajuster les dimensions d'affichage des organigrammes. Le *facteur d'agrandissement* permet ainsi de redimensionner l'organigramme affiché de 25% à 200% de sa taille normale. Il est ainsi possible d'avoir une vue d'ensemble d'un grand organigramme ou de focaliser l'affichage sur une petite section de celui-ci.

Le facteur d'agrandissement de l'éditeur graphique est ajustable via la [barre de menu](#) ainsi que le via les [commandes de l'éditeur](#). Le [panneau de statut](#) affiche le facteur d'agrandissement courant.

2.4.6 Surligne d'instructions en exécution pas-à-pas

Lors de l'[exécution pas-à-pas](#) d'un projet, l'éditeur graphique surligne la prochaine instruction de l'algorithme à être exécutée (Figure 2-22) ainsi que les [points d'arrêts](#) actifs (Figure 2-24) :

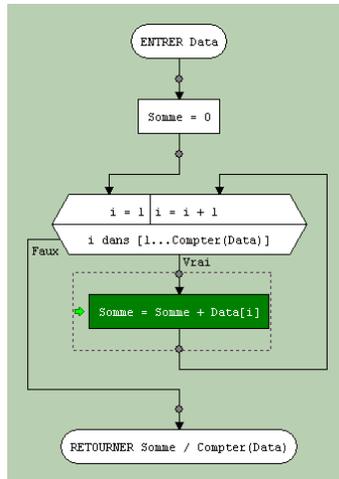


Figure 2-22 : Prochaine instruction à exécuter

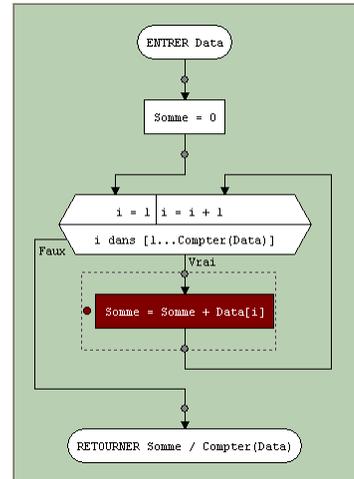


Figure 2-23 : Point d'arrêt surligné

Les couleurs de surligne dans l'éditeur graphique sont [configurables](#).

2.4.7 Configuration de l'éditeur graphique

Seules les couleurs d'affichage et la police de caractères sont configurables dans l'éditeur graphique. Pour plus d'information consultez la section portant sur la [configuration des éditeurs de LARP](#).

2.4.8 Commandes d'édition de l'éditeur graphique

Les commandes d'édition ainsi que celles de gestion de projet sont accessibles via :

- la [barre de menu](#),
- les [menus contextuels](#),
- le [panneau de contrôle](#),
- le [clavier](#), et
- la [souris](#).

De plus, l'éditeur graphique est généralement exploité avec le [panneau de modèles](#) afin de concevoir des organigrammes par glisser-déposer.

2.4.8.1 Commandes de l'éditeur graphique accessibles via les menus

On retrouve dans *LARP* deux types de menus :

1. la [barre de menu](#), affichée au haut de la fenêtre principale de l'[environnement de développement](#), et
2. les menus contextuels, associés à divers éléments de l'environnement de développement et accessibles via le bouton droit de la souris lorsque celle-ci est cliquée sur l'élément d'interface visé.

L'éditeur graphique répond à diverses commandes de la barre de menu (voir la section portant sur la [barre de menu](#)) ainsi qu'à celles des menus contextuels associés aux instructions et aux nœuds d'insertion dans l'organigramme en cours d'édition. Ces menus comprennent, entre autres :

- des commandes d'[édition d'organigramme](#), telle que l'insertion d'instructions via un nœud (Figure 2-20),
- des commandes de [manipulation d'instructions](#), telles que l'édition, le changement d'orientation et la transformation, et
- des commandes de gestion du [presse-papiers](#) et de [recherche et remplacement](#).

Pour une description détaillée des commandes accessibles via les menus, consultez les sections décrivant la [barre de menu](#), le [panneau de contrôle](#) et l'[édition d'organigrammes](#).

2.4.8.2 Commandes de l'éditeur graphique accessibles via le clavier

En plus des [commandes accessibles via les menus](#), l'éditeur graphique de *LARP* répond à diverses commandes accessibles via une touche ou une combinaison de touches du clavier :

Touches	Description
<End>	Sélectionne la dernière instruction de l'organigramme.
<Home>	Sélectionne la première instruction de l'organigramme.
<Enter>	Affiche le menu contextuel associé à l'instruction sélectionnée ou au nœud d'insertion sélectionné.
	Supprime l'instruction d'organigramme sélectionnée.
+	Augmente le facteur d'agrandissement.
-	Réduit le facteur d'agrandissement.
<Left> <Up>	Sélectionne l'instruction ou le nœud d'insertion précèdent la sélection courante, en fonction des branchements associés à celle-ci.
<Right> <Down>	Sélectionne la prochaine instruction ou le prochain nœud d'insertion en fonction des branchements associés à la sélection courante.
<Shift>+<Left> <Right> <Up> <Down>	Mêmes fonctionnalités que les touches <Left>, <Right>, <Up> et <Down>, mais tente de trouver une sélection alternative.
<Ctrl>+<Left> <Right> <Up> <Down>	Fonctionnalité similaire à l'utilisation de la touche <Shift>.

Tableau 2-5 : Commandes de l'éditeur graphique accessibles via le clavier

Notez que l'emploi des touches <Shift> ou <Ctrl> avec les touches directionnelles (<Left>, <Right>, <Up> ou <Down>) permet une sélection alternative à l'emploi des touches directionnelles seules. Puisque l'éditeur graphique doit déterminer quelle instruction ou nœud d'insertion sélectionner en fonction de la touche pressée, il peut à l'occasion effectuer une sélection différente de celle anticipée par l'utilisateur. L'emploi des touches <Shift> ou <Ctrl> permet à l'utilisateur de tenter de corriger la situation.

En plus des commandes ci-dessus, *LARP* répond aussi aux touches accélératrices associées à la [barre de menu](#).

2.4.8.3 Contrôle de l'éditeur graphique via la souris

L'éditeur graphique de l'[environnement de développement](#) répond de façon standard aux interactions via la souris :

- Un clic du bouton gauche de la souris sur une instruction de l'organigramme ou sur un nœud d'insertion sélectionne l'élément visé.
- Un double clic du bouton gauche sur une instruction ouvre la fenêtre d'édition du contenu de l'instruction (voir Figure 2-21).
- Un clic du bouton droit de la souris sur une instruction ou sur un nœud d'insertion affiche le menu contextuel associé à l'élément visé.

De plus, la souris peut être employée avec le [panneau de modèles](#) pour insérer des instructions dans l'organigramme via glisser-déposer. Pour se faire il faut sélectionner un modèle dans panneau de modèles, le glisser au nœud d'insertion où l'instruction doit être insérée dans l'organigramme affiché dans l'éditeur graphique, et relâcher le bouton de la souris. L'instruction insérée peut ensuite être [modifiée](#) afin de l'adapter aux besoins de l'algorithme.

La souris peut aussi être employée pour déplacer une instruction de l'organigramme édité à une autre position dans l'organigramme. Il suffit de sélectionner l'instruction de l'organigramme à déplacer en cliquant sur celle-ci avec la souris, glisser l'instruction sélectionnée vers le nœud d'insertion correspondant à son nouvel emplacement dans l'organigramme, et finalement déposer l'instruction glissée sur le nœud visé. L'éditeur graphique déplace alors l'instruction sélectionnée vers sa nouvelle position.

Lorsqu'un modèle ou une instruction est glissé vers un nœud d'insertion de l'organigramme, la couleur résultante du nœud visé indique si l'instruction glissée peut y être déposée : un nœud **rouge** indique qu'il est logiquement erroné d'y déposer l'instruction ou le modèle glissé, un nœud **vert** autorisant le dépôt.

2.5 Compilation et exécution

La compilation d'un projet *LARP* consiste à vérifier la conformité des pseudo-codes et organigrammes du projet à la syntaxe permise par *LARP*, puis à transformer ceux-ci en une forme exécutable. Si des erreurs de syntaxe sont détectées dans un module lors de la compilation, des [messages d'erreur](#) sont affichés dans le [panneau de messages](#) et aucun code exécutable n'est produit.

Lors de la compilation d'un projet, *LARP* affiche une fenêtre indiquant l'état de la compilation (Figure 2-24). Cette fenêtre indique le nom du projet en cours de compilation, ainsi que le résultat de la compilation (i.e. si elle fut un succès ou non). Les compteurs indiquent le nombre de messages d'avertissement et d'erreur détectés à la compilation et affichés dans le [panneau de messages](#).

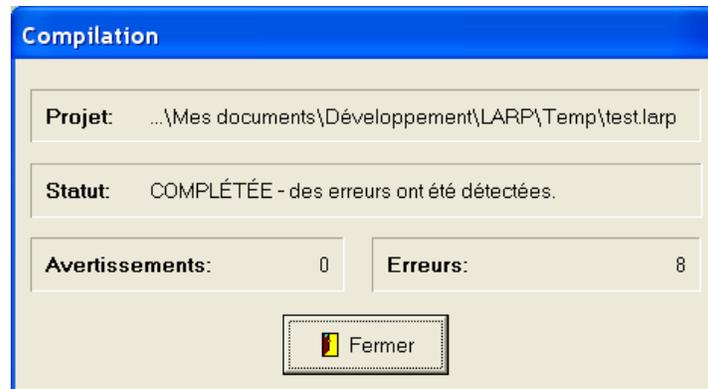


Figure 2-24 : Fenêtre de compilation

Lorsqu'il n'y a pas d'erreur détectée et le projet est compilé avec succès, il peut être exécuté via la [console d'exécution](#).

Les commandes de compilation et d'exécution sont accessibles via la [barre de menu](#) et le [panneau de contrôle](#).

2.5.1 Exécution d'un projet

Lorsqu'un projet est compilé avec succès, c'est-à-dire sans que des erreurs de syntaxe aient été détectées, il peut être exécuté via la [console d'exécution](#). Si l'utilisateur tente d'exécuter un projet n'ayant pas préalablement été compilé, LARP compile automatiquement celui-ci avant de l'exécuter.

Durant l'exécution d'algorithmes, les entrées/sorties sont dirigées vers la console d'exécution. Si une erreur de logique est détectée, un [message d'avertissement ou d'erreur](#) est immédiatement affiché dans le [panneau de messages](#) et, si l'erreur est fatale, l'exécution est interrompue.

L'utilisateur peut en tout temps interrompre l'exécution d'un projet en fermant la console d'exécution.

2.5.2 Exécution pas-à-pas

L'exécution pas-à-pas permet de contrôler l'exécution d'un algorithme. Dans ce contexte, l'utilisateur peut entre autres momentanément interrompre l'exécution de l'algorithme, exécuter celui-ci une instruction à la fois, suivre l'évolution du contenu des variables et animer l'exécution de certaines instructions.

L'exécution pas-à-pas est démarrée via la commande **Exécuter » Exécuter pas-à-pas** de la [barre de menu](#). Le [panneau de contrôle](#) dispose aussi d'un bouton permettant d'activer l'exécution pas-à-pas.

2.5.2.1 Interface de l'exécution pas-à-pas

La Figure 2-25 présente la *fenêtre d'exécution pas-à-pas* et ses différents éléments. Cette fenêtre est affichée en avant plan de la [console d'exécution](#) lorsque l'exécution pas-à-pas est activée (via la commande **Exécuter » Exécuter pas-à-pas** de la [barre de menu](#) ou via le bouton correspondant du [panneau de contrôle](#) de la fenêtre principale de l'[environnement de développement](#)).

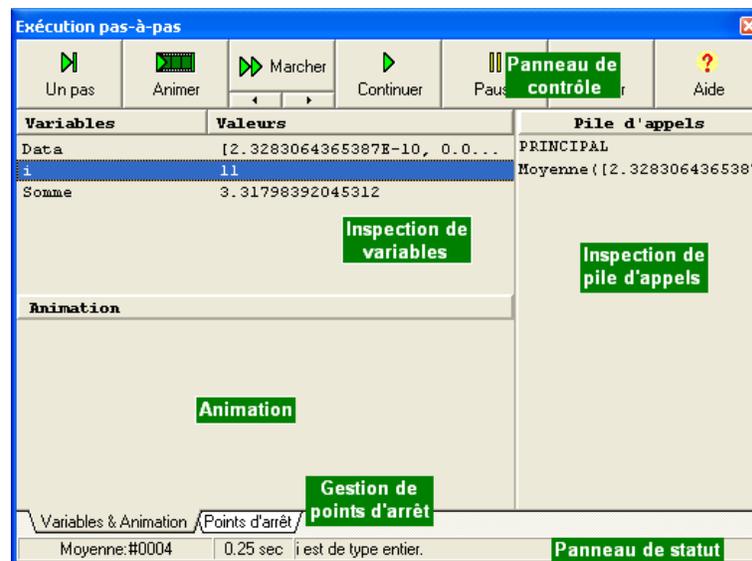


Figure 2-25 : Éléments de la fenêtre d'exécution pas-à-pas

Les éléments de la fenêtre d'exécution pas-à-pas sont :

1. Le **panneau de contrôle** : à ne pas confondre avec le [panneau de contrôle de la fenêtre principale de l'environnement de développement](#), celui de la fenêtre d'exécution pas-à-pas permet de contrôler l'exécution de l'algorithme. Chaque bouton du panneau offre une fonctionnalité distincte :

-  Exécute la prochaine instruction de l'algorithme ([mode pas-à-pas](#)). L'exécution de l'algorithme est automatiquement suspendue après l'exécution d'une seule instruction. Voir les [modes d'exécution pas-à-pas](#) pour plus d'information.
-  Active le mode d'exécution en animation de l'algorithme. La vitesse d'animation est contrôlée via les boutons sous le bouton de **Marche**. Voir la section portant sur l'[animation](#) d'instructions pour plus d'information.
-  Active le mode d'exécution en marche de l'algorithme. Les deux boutons sous celui de marche permettent de contrôler la vitesse de marche. L'exécution en marche peut être suspendue en tout temps via le bouton **Pause**. Voir les [modes d'exécution pas-à-pas](#) pour plus d'information.
-  Active le mode d'exécution en continu de l'algorithme. L'exécution en continu peut être suspendue en tout temps via le bouton **Pause**. Voir les [modes d'exécution pas-à-pas](#) pour plus d'information.
-  Suspend temporairement l'exécution en marche ou l'exécution en continu de l'algorithme. Voir les [modes d'exécution pas-à-pas](#) pour plus d'information.
-  Interrompt définitivement l'exécution pas-à-pas de l'algorithme. Ce bouton ferme la fenêtre d'exécution pas-à-pas ainsi que la [console d'exécution](#).
-  Accède à l'[aide en ligne](#) associé à l'exécution pas-à-pas.

L'exécution pas-à-pas de l'algorithme peut être interrompu en tout temps en fermant la fenêtre d'exécution pas-à-pas ou en pressant le bouton **Arrêter** du panneau de contrôle.

2. Le **panneau de statut** : Le panneau de statut de la fenêtre d'exécution pas-à-pas, à ne pas confondre avec celui de la fenêtre principale de l'environnement de développement, affiche de l'information pertinente à l'exécution en cours :

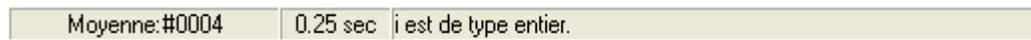


Figure 2-26 : Panneau de statut de la fenêtre d'exécution pas-à-pas

Les trois sections du panneau affichent l'information suivante :

- La première section indique les coordonnées de la prochaine instruction à être exécutée (nom du module et numéro de l'instruction).
 - La deuxième section indique la vitesse d'exécution pas-à-pas et d'animation en secondes. Cette vitesse est ajustable via les boutons flèches sous celui de **Marche** du panneau de contrôle.
 - La troisième section indique de l'information sur la variable, le point d'arrêt ou l'élément de pile d'appels sélectionné.
3. Le **panneau d'inspection des variables** : affiche la liste des variables définies dans le module d'algorithme en cours d'exécution, ainsi que la valeur de chacune des variables. Le type d'une variable est automatiquement affichée dans le panneau de statut lorsque celle-ci est sélectionnée dans le panneau d'inspection des variables.

Pour plus d'information sur le panneau d'inspection des variables, consultez la section 2.5.2.3.

4. Le **panneau d'inspection de la pile d'appels** : affiche l'état de la pile d'appels. La pile d'appels consiste en la séquence d'appels de modules exécutés depuis le module principal jusqu'au l'instruction en cours d'exécution. Le panneau affiche le nom de chaque module dans la pile, ainsi que la valeur des paramètres d'appels de chacun. L'appel de module est automatiquement affichée en entier dans le panneau de statut lorsque celle-ci est sélectionnée dans le panneau d'inspection de la pile d'appels.

Pour plus d'information sur le panneau d'inspection de pile d'appels, consultez la section 2.5.2.4.

5. Le **panneau d'animation** : affiche l'animation d'exécution d'instructions. Lorsque l'animation d'une instruction est activée, ce panneau présente une animation vidéo du processus d'évaluation des différents composants de l'instruction.

Pour plus d'information sur le panneau d'animation d'instructions, consultez la section 2.5.2.6.

6. Le **panneau de gestion des points d'arrêt** : affiche les points d'arrêts actuellement défini dans l'algorithme en cours d'exécution, et permet de désactiver ceux-ci au besoin.

Pour plus d'information sur la gestion des points d'arrêt, consultez la section 2.5.2.5.

Les prochaines sections décrivent en détails les fonctionnalités des principaux éléments de la fenêtre d'exécution pas-à-pas.

2.5.2.2 Modes d'exécution pas-à-pas

Un projet peut être exécuté pas-à-pas selon trois modes d'exécution :

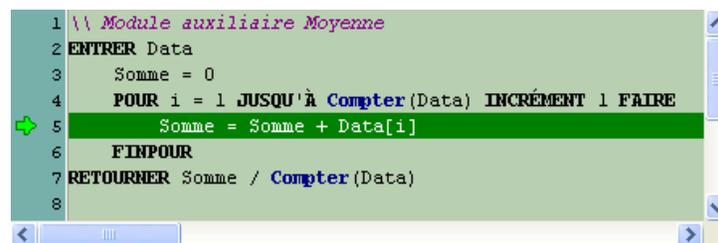
1. **Par pas** : une seule instruction d'algorithme est exécutée. L'utilisateur peut par la suite exploiter la [fenêtre d'exécution pas-à-pas](#) pour inspecter le contenu des variables et de la pile d'appel. L'[animation](#) peut être activée durant l'exécution de l'instruction. Le panneau de contrôle de la fenêtre d'exécution pas-à-pas dispose du bouton **Un pas** permettant d'activer l'exécution de la prochaine instruction d'algorithme. Le bouton **Animer** fait de même tout en animant l'exécution de l'instruction.
2. **En marche** : le projet est exécuté au ralenti, une pause de durée ajustable étant insérée entre l'exécution de chaque instruction. Le panneau de contrôle de la fenêtre d'exécution pas-à-pas dispose du bouton **Marcher** activant le mode d'exécution en marche de l'algorithme. Les boutons fléchés sous le bouton **Marcher** permettent de modifier la vitesse de marche. Celle-ci est affichée (en secondes de délai inséré entre l'exécution d'instructions) dans le panneau de statut de la fenêtre d'exécution pas-à-pas.

L'exécution en marche est interrompue dès que la fin du [module principal](#) est atteinte ou qu'un [point d'arrêt](#) est rencontré. Le bouton **Pause** du panneau de contrôle permet en tout temps de suspendre l'exécution en marche.

3. **En continue** : ce mode d'exécution est semblable à celui en marche, mais aucun délai n'est inséré entre chaque exécution d'instruction. L'exécution en continue est interrompue dès que la fin du module principal est atteinte ou qu'un [point d'arrêt](#) est rencontré. Le bouton **Pause** du panneau de contrôle permet en tout temps de suspendre l'exécution.

Il est à noter que l'exécution en continue via la fenêtre d'exécution pas-à-pas est considérablement plus lente que l'[exécution conventionnelle](#) (sans exploiter la fenêtre d'exécution pas-à-pas), cette dernière ignorant les points d'arrêts rencontrés. La gestion des points d'arrêts en exécution pas-à-pas ainsi que l'affichage du contenu de la fenêtre d'exécution pas-à-pas exigent plus de ressources de calculs de l'ordinateur.

Lors de l'exécution en mode par pas ou en mode marche d'un projet, l'[éditeur textuel](#) et l'[éditeur graphique](#) surlignent la prochaine instruction de l'algorithme en attente d'exécution :



```
1  \\ Module auxiliaire Moyenne
2  ENTRER Data
3      Somme = 0
4  POUR i = 1 JUSQU'À Compter(Data) INCRÉMENT 1 FAIRE
5  Somme = Somme + Data[i]
6  FINPOUR
7  RETOURNER Somme / Compter(Data)
8
```

Figure 2-27 : Prochaine instruction à exécuter (pas-à-pas) dans l'éditeur textuel

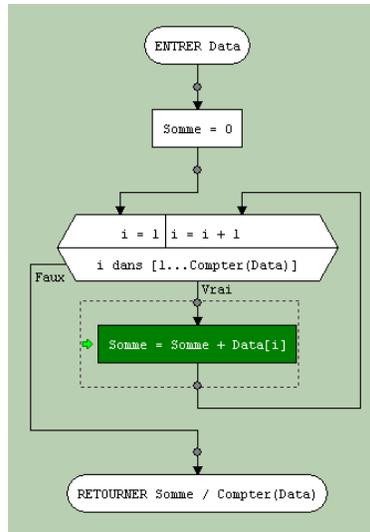


Figure 2-28 : Prochaine instruction à exécuter (pas-à-pas) dans l'éditeur graphique

2.5.2.3 Inspection des variables

Le panneau d'inspection des variables (Figure 2-29) affiche le contenu de chaque variable du module en exécution durant les pauses lors de l'exécution pas-à-pas.

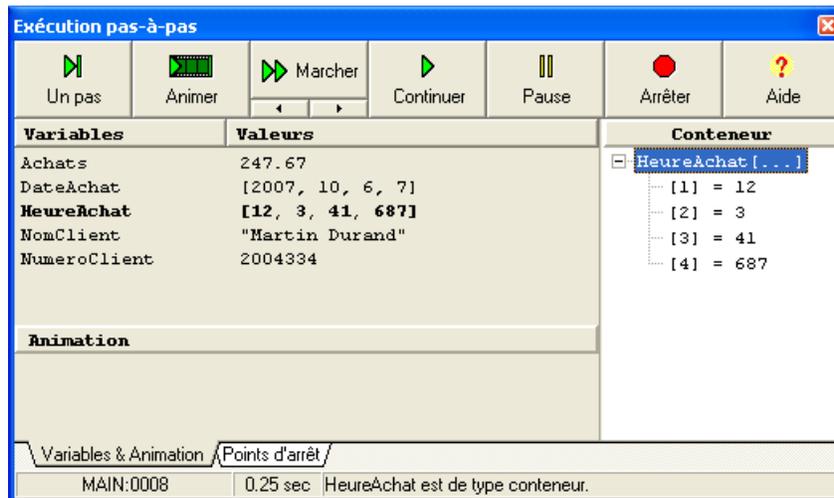


Figure 2-29 : Panneau d'inspection des variables

Le panneau affiche chaque variable définie dans le module en cours d'exécution ainsi que sa valeur. Lorsque la dernière instruction exécutée modifie une valeur de variables, celle-ci est affichée en caractères gras dans le panneau.

Lorsqu'une variable est sélectionnée dans le panneau, le type de son contenu est affiché dans le [panneau de statut](#). Si la variable sélectionnée est un [conteneur](#), le [panneau d'inspection de la pile d'appels](#) est remplacé temporairement par un panneau, intitulé **Conteneur**, permettant d'inspecter les éléments du conteneur en question.

Le suivi de l'évolution du contenu des variables lors de l'exécution pas-à-pas facilite l'identification et l'élimination de bogues dans les algorithmes d'un projet LARP.

2.5.2.4 Inspection de la pile d'appels

La *pile d'appels* d'un projet *LARP* en exécution consiste en la séquence d'appels de modules exécutés depuis le **module principal** jusqu'à l'instruction en cours d'exécution. La fenêtre d'exécution pas-à-pas affiche à droite un panneau énumérant le nom de chaque module dans la pile (en commençant par le module principal), ainsi que la valeur des **paramètres** d'appels de chacun :

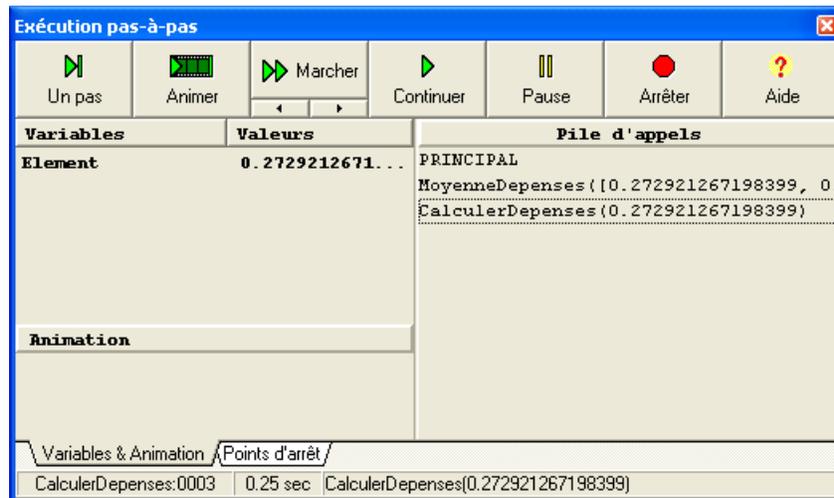


Figure 2-30 : Panneau d'inspection de la pile d'appels

Dans la Figure 2-30, l'exécution pas-à-pas est en attente d'exécuter l'instruction 0003 du module `CalculerDepenses`, celui-ci ayant été invoqué par le module `MoyenneDepenses` qui fut invoqué par le module principal (`PRINCIPAL`). Le panneau d'inspection de la pile d'appels affiche la valeur de chaque paramètre d'appel.

L'appel de module est automatiquement affichée en entier dans le **panneau de statut** lorsque celle-ci est sélectionnée dans le panneau d'inspection de la pile d'appels.

2.5.2.5 Points d'arrêt

Le panneau d'inspection des points d'arrêt (Figure 2-31) énumère les points d'arrêt actifs durant l'exécution pas-à-pas du projet *LARP*. Un *point d'arrêt* indique une instruction de module où l'exécution pas-à-pas en **mode marche** ou **en continue** doit s'interrompre afin de permettre à l'utilisateur d'**inspecter les variables**, d'**inspecter la pile d'appel**, de changer de **mode d'exécution pas-à-pas** ou d'activer l'**animation**.

Le panneau d'inspection des points d'arrêt affiche tous les points d'arrêt définis et surligne (en rouge dans la Figure 2-31) le point d'arrêt ayant interrompu l'exécution pas-à-pas :

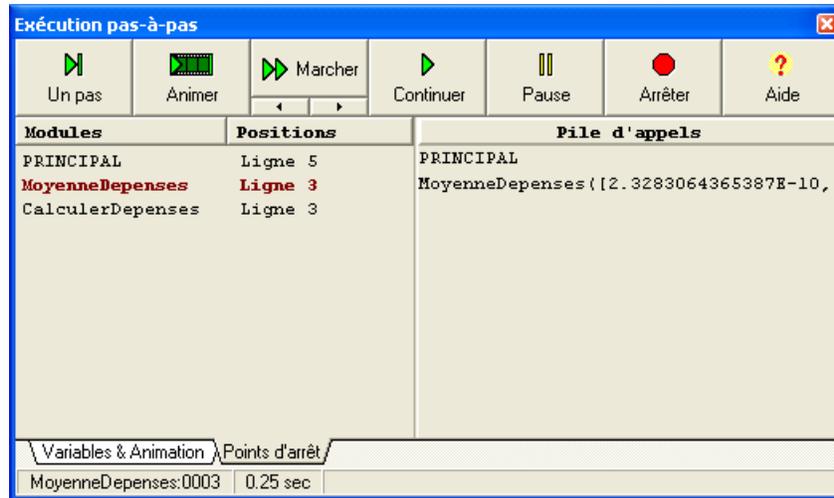


Figure 2-31 : Panneau d'inspection des points d'arrêt

Le menu contextuel du panneau d'inspection des points d'arrêt, accessible en cliquant sur le panneau avec le bouton droit de la souris, permet de localiser l'emplacement d'un point d'arrêt dans le projet, et de désactiver les points d'arrêt.

L'éditeur [textuel](#) et l'éditeur [graphique](#) de l'environnement de développement surlignent les points d'arrêt associés aux instructions d'un module :

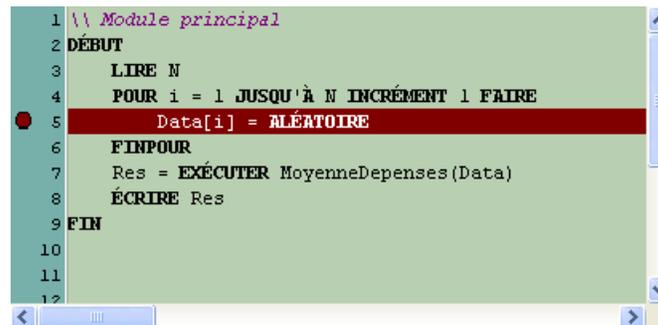


Figure 2-32 : Point d'arrêt surligné dans l'éditeur textuel

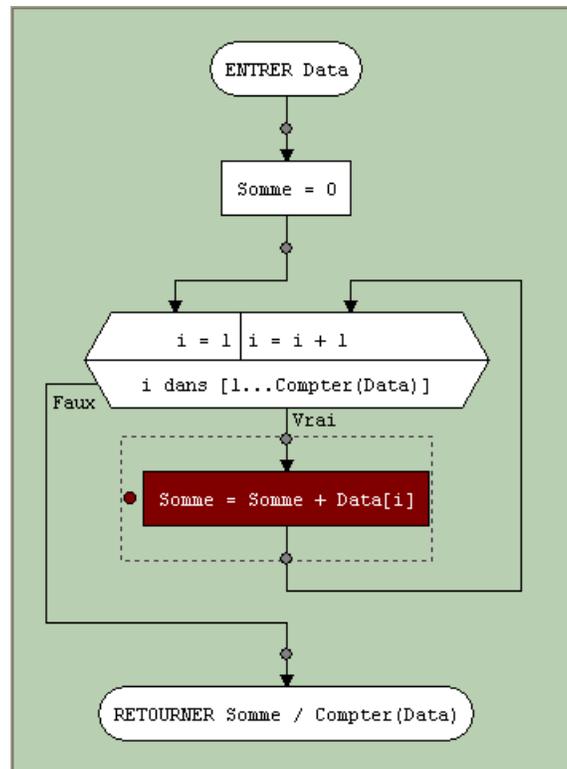


Figure 2-33 : Point d'arrêt surligné dans l'éditeur graphique

Les menus contextuels des éditeurs ainsi que la [barre de menu](#) donnent accès aux commandes permettant de créer et supprimer des points d'arrêt.

2.5.2.6 Animation

Le panneau d'animation de la fenêtre d'exécution pas-à-pas (Figure 2-34) permet de visualiser le processus d'évaluation d'une instruction d'algorithme. L'animation aide ainsi à comprendre les étapes d'exécution d'une instruction d'algorithme en fonction de ses composants. L'animation vise essentiellement à atteindre un objectif pédagogique particulier : aider l'apprenant à comprendre l'importance des priorités dans l'évaluation des [opérateurs arithmétiques](#), des [opérateurs relationnels](#) et des [opérateurs logiques](#).

La Figure 2-34 présente en exemple l'animation d'une [instruction d'affectation](#). L'animation décompose le processus d'évaluation de l'équation mathématique à droite du symbole d'affectation (=) selon les [priorités d'évaluation des opérateurs arithmétiques](#). Ainsi 2^k (qui donne 4.59479341889914) est évalué avant la multiplication, cette dernière résultant à 16.8628918513565. L'addition est effectuée par la suite, et son résultat est 37.8628918513565. Cette dernière valeur est finalement affectée à la variable z . À chaque étape de décomposition du processus d'évaluation, le panneau d'animation affiche les résultats intermédiaires de l'application des opérateurs arithmétiques (par exemple l'évaluation de la variable x donne 45, dont la racine carré donne 6.70820393249937).

La Figure 2-35 présente un exemple d'évaluation d'une [condition](#) de structure conditionnelle ([la structure SI](#)). La décomposition de l'évaluation de la condition permet d'observer que celle-ci est affirmative (i.e. le résultat est `VRAI`).

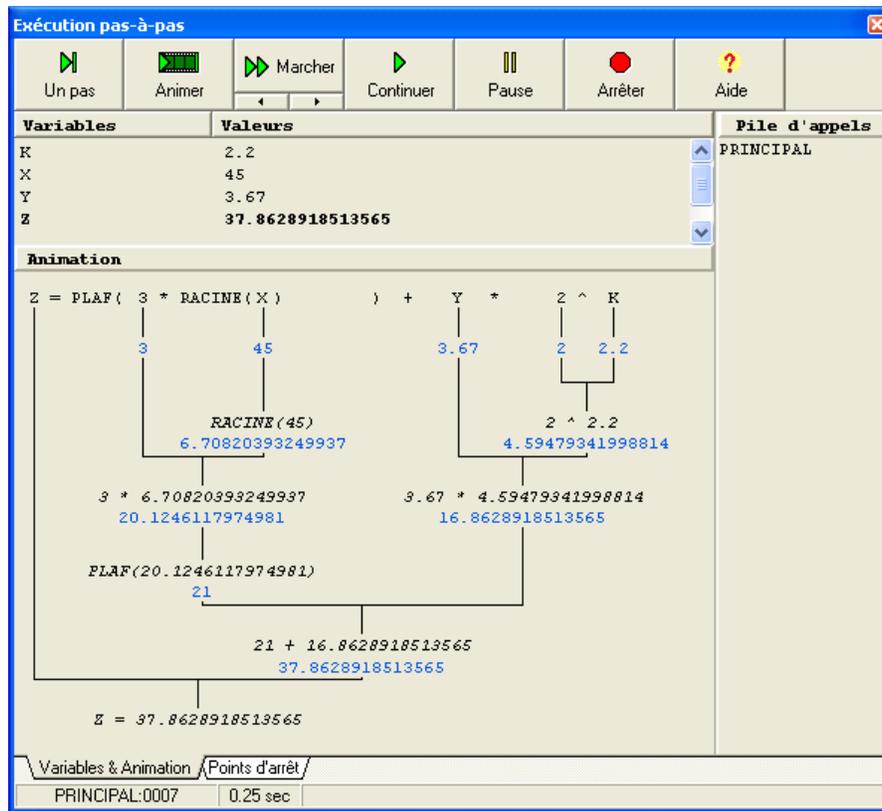


Figure 2-34 : Panneau d'animation d'instruction

Pour animer l'exécution de la prochaine instruction lors de l'exécution pas-à-pas, il suffit de presser le bouton **Animer** plutôt que le bouton **Un pas**. Le panneau d'animation est automatiquement activé et le processus d'évaluation de l'instruction est animé avec des pauses entre l'évaluation de chaque composant, et ce jusqu'à ce que l'évaluation de l'instruction soit complétée. La durée des pauses entre l'évaluation de chaque composant de l'instruction correspond à la pause insérée entre l'exécution des instructions en mode marche. Les boutons fléchés sous le bouton **Marcher** permettent de modifier la vitesse d'animation (et de marche). Celle-ci est affichée (en secondes de délai inséré entre l'exécution de composants) dans le **panneau de statut** de la fenêtre d'exécution pas-à-pas.

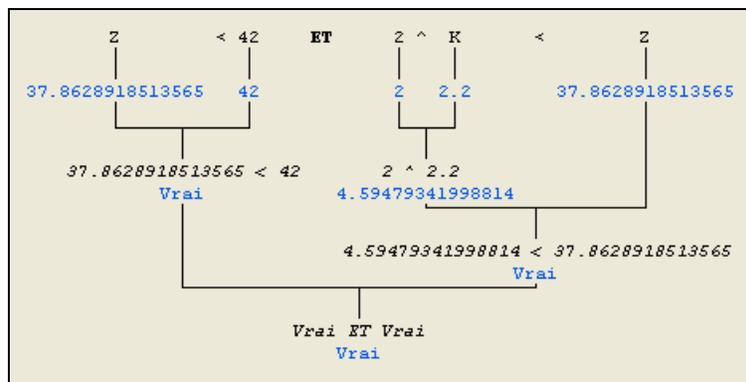


Figure 2-35 : Animation de l'évaluation d'une condition

Voici quelques notes importantes à propos de l'animation :

- Une fois l'animation d'une instruction débutée, le processus d'animation ne peut pas être interrompu, et ce jusqu'à la fin de l'animation de l'instruction.
- La vitesse d'animation peut être modifiée en tout temps (via les boutons fléchés sous le bouton **Marcher**), même durant une animation.
- L'animation est limitée à l'évaluation d'opérateurs arithmétiques, relationnels et logiques. L'évaluation des autres composants d'une instruction d'algorithme ne peuvent généralement pas être animés.
- Le panneau d'animation affiche toujours l'instruction à animer sous forme pseudo-code, même si le **module** contenant l'instruction est sous forme d'organigramme.

2.5.3 Sauvegarde de sécurité

Puisque la technologie n'est pas infaillible (ainsi que ne le sont pas les talents en programmation de l'auteur de *LARP*), il peut arriver à l'occasion que *LARP* cesse abruptement de fonctionner sans avoir donné l'opportunité à l'utilisateur de sauvegarder les dernières modifications à son projet.

Pour minimiser les pertes de travail advenant de un tel incident, *LARP* effectue périodiquement (par défaut, à toutes les 10 minutes) une sauvegarde de sécurité du projet édité dans un fichier temporaire. De plus, puisque les probabilités de défaillance de *LARP* sont accrues durant l'exécution d'un projet, *LARP* effectue une sauvegarde de sécurité avant chaque exécution du projet.

Si, comme c'est généralement le cas, tout se déroule normalement et l'utilisateur sauvegarde son projet selon la procédure normale (via la **barre de menu** ou le **panneau de contrôle**), la sauvegarde de sécurité du projet est automatiquement détruite. Si par contre *LARP* crashe avant que l'utilisateur ait eu l'opportunité de sauvegarder son travail, la sauvegarde de sécurité demeure intacte, n'étant pas détruite.

À chaque démarrage, *LARP* recherche la présence d'une sauvegarde de sécurité antérieure. Si une telle sauvegarde est localisée, *LARP* propose à l'utilisateur de recharger cette sauvegarde afin de récupérer le projet modifié. Si l'utilisateur refuse cette option, le projet n'est pas rechargé dans *LARP*. Dans un cas comme dans l'autre, la sauvegarde de sécurité est ensuite détruite.

2.5.4 Avertissements et erreurs

Lors de la compilation et de l'exécution d'un projet, *LARP* affiche dans le **panneau de messages** une brève description des erreurs rencontrées :

- **Lors de la compilation** : les erreurs de syntaxe rencontrées dans les pseudo-codes et organigrammes du projet sont identifiées.
- **Lors de l'exécution** : les erreurs de logique rencontrées dans les pseudo-codes et organigrammes du projet sont identifiées.

LARP affiche aussi à l'occasion des messages d'avertissement. Ces messages identifient généralement des erreurs de logique qui ne sont pas fatales, n'empêchant pas l'algorithme d'être exécuté mais pouvant rendre son comportement imprévisible lors de l'exécution.

Chaque message d'avertissement ou d'erreur affiché dans le panneau de messages comprend généralement les informations suivantes :

1. le type de message (avertissement ou erreur);
2. la position de l'anomalie dans le projet (le nom du module et la position de l'instruction dans le module où est localisée l'anomalie);
3. une brève description de l'anomalie;
4. optionnellement, un numéro de référence permettant d'obtenir plus d'information via l'[aide en ligne](#) de *LARP*.

Pour localiser l'emplacement de l'erreur ou de l'avertissement dans le projet, l'utilisateur peut double cliquer sur le message visé dans le panneau de messages. L'éditeur approprié (soit l'[éditeur textuel](#) ou l'[éditeur graphique](#), dépendant du type de module) affiche alors le module contenant l'erreur et l'instruction erronée y est identifiée. Pour obtenir de l'information supplémentaire via l'aide en ligne, il faut cliquer sur le message avec le bouton droit de la souris afin d'accéder au menu contextuel du panneau de message, qui permet d'invoquer l'aide en ligne.

Pour plus d'information sur les messages d'avertissement et d'erreur, consultez l'[Annexe E](#). Vous pouvez aussi consulter les sections portant sur le [panneau de messages](#) et sur l'[aide en ligne](#) pour en connaître plus sur la gestion des messages d'avertissement et d'erreur dans *LARP*.

2.6 Configuration de *LARP*

L'[environnement de développement](#) de *LARP* est configurable du point de vue :

- des [fonctionnalités d'édition](#),
- de l'[exécution d'algorithmes](#),
- de la [gestion générale](#), et
- des [couleurs d'affichage](#).

La configuration courante est sauvegardée dans les registres de l'ordinateur à la fermeture de *LARP*. Cette configuration est automatiquement réactivée au prochain démarrage du logiciel.

Note : Il est fortement déconseillé d'altérer la configuration de *LARP* directement via les registres de l'ordinateur (avec un utilitaire tel **regedit**). Une corruption des registres peut irrémédiablement endommager le système d'exploitation de l'ordinateur. Il est recommandé d'utiliser les fonctionnalités de configuration intégrées à *LARP* pour modifier la configuration du logiciel.

2.6.1 Configuration générale

La *fenêtre de configuration générale* (Figure 2-36) de *LARP* permet de configurer différents éléments de l'[environnement de développement](#) ainsi que le processus d'exécution d'algorithmes.

Cette fenêtre est accessible via la [barre de menu](#), sous l'item **Options » Générales** :

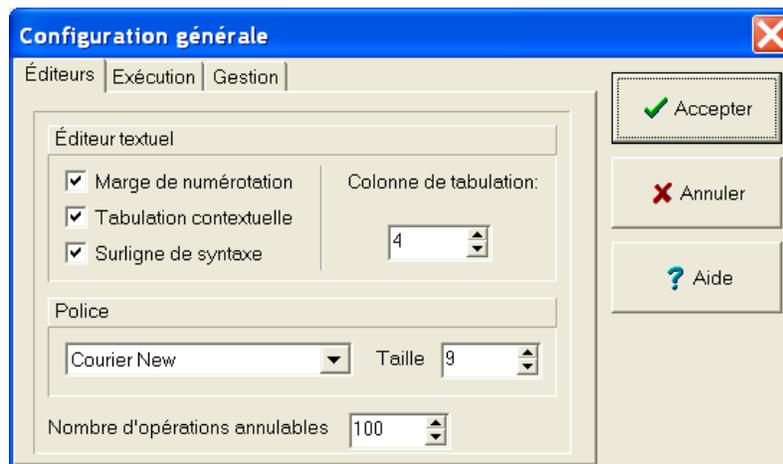


Figure 2-36 : Configuration générale

Les options de configuration sont regroupées sous trois rubriques (accessibles via les onglets au haut de la fenêtre) :

1. **Éditeurs** : options de configuration des [éditeurs](#) de *LARP*.
2. **Exécution** : options de configuration de la [console d'exécution](#) de *LARP*.
3. **Gestion** : options de configuration du [mode super-utilisateur](#) et du [système de mise à jour](#) intégré à *LARP*.

2.6.1.1 Configuration des éditeurs

Les options de configuration sous la rubrique **Éditeurs** permettent de modifier les attributs d'édition.

Les attributs suivants ont trait aux deux éditeurs (l'[éditeur textuel](#) et l'[éditeur graphique](#)) :

- **Police** : ces attributs indiquent la police de caractères ainsi que la taille des caractères employés dans les éditeurs. Il est recommandé d'employer une police à taille fixe, telle *Courier New*, car une telle police facilite l'alignement des caractères dans les instructions d'affichage.
- **Nombre d'opérations annulables** : cette valeur correspond à la taille du tampon de sauvegarde des opérations d'édition. Ce tampon permet d'annuler les plus récentes opérations d'édition effectuées par l'utilisateur. La valeur spécifiée correspond au nombre d'opérations enregistrées et annulables.

Les attributs suivants ont trait à l'[éditeur textuel](#) exclusivement :

- **Tabulation contextuelle** : lorsque la tabulation contextuelle est activée, la touche de tabulation insère des espaces au curseur de sorte que ce dernier soit déplacé vis-à-vis le prochain caractère de la ligne précédent celle au curseur. La tabulation contextuelle facilite l'alignement des instructions de pseudo-code sur des lignes successives.

Lorsque la tabulation contextuelle est désactivée, la touche de tabulation insère un caractère de tabulation au curseur.

- **Surligne de syntaxe** : lorsque la surligne syntaxique est activée, les mots réservés et autres attributs d'un pseudo-code sont affichés dans l'éditeur à l'aide de [couleurs distinctes](#). La surligne permet à l'utilisateur d'identifier rapidement les différents éléments d'un pseudo-code.
- **Marge de numérotation** : lorsque la marge de numérotation est activée, une marge apparaît à gauche du panneau d'édition de l'éditeur textuel. Cette marge affiche le numéro des lignes de texte du document édité, ainsi que les signets.
- **Colonne de tabulation** : cette valeur correspond à la taille d'une tabulation conventionnelle (i.e. lorsque la tabulation contextuelle est désactivée). Par exemple, si la valeur spécifiée est 4, une pression de la touche de tabulation déplacera le curseur à la prochaine colonne multiple de 4 (i.e. 4, 8, 12, 16, 20, ...).

2.6.1.2 Configuration de la console d'exécution

Les options de configuration sous la rubrique **Exécution** (Figure 2-37) permettent de configurer les paramètres d'exécution de projets LARP.

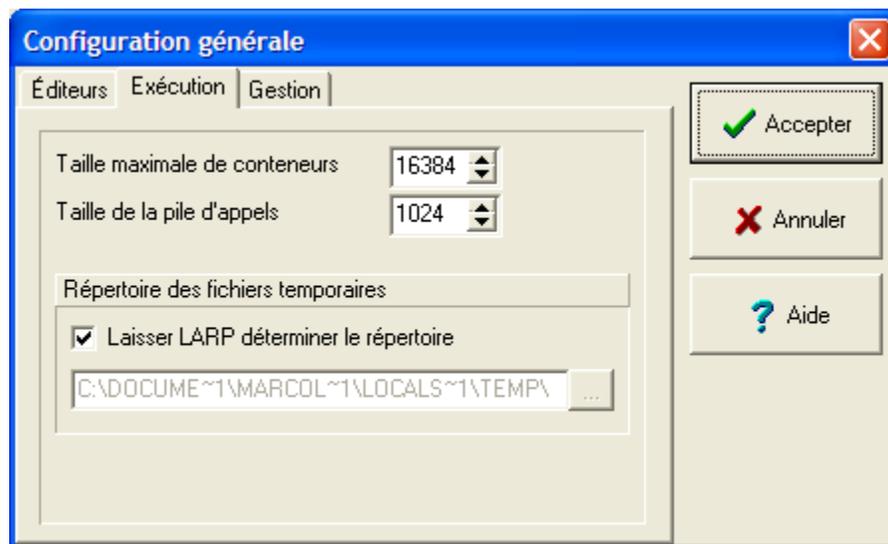


Figure 2-37 : Configuration de l'exécution d'algorithmes

Les paramètres configurables sont :

- **Taille maximale de conteneurs** : ce paramètre indique l'index maximal que peut avoir un élément dans un [conteneur](#) (l'index minimal étant fixé à 1). Cette limite évite la création de conteneurs de taille excessivement grande, ce qui peut engendrer un manque de mémoire vive de l'ordinateur. La création de conteneurs trop grands est généralement causée par une erreur de logique dans l'algorithme.
- **Taille de la pile d'appels** : fixe le nombre maximum d'invocations de modules pouvant être imbriquées lors de l'exécution de l'algorithme (des invocations sont imbriquées lorsqu'un module invoque un second module, qui invoque un troisième module, qui en invoque un quatrième, ...). Cette limite est essentiellement utilisée pour interrompre les invocations récursives infinies. Voir la section portant sur la [récursivité](#) pour plus d'information.

- **Répertoire des fichiers temporaires** : indique le chemin et le nom du répertoire où sont stockés les divers fichiers temporaires créés par *LARP* lors de l'exécution d'algorithmes. Ces fichiers incluent ceux servant à la gestion de [tampons d'entrées/sorties](#), ainsi que les fichiers de [sauvegarde de sécurité](#) (en cas d'erreur fatal de *LARP*).

Il est recommandé de laisser à *LARP* la tâche de sélectionner un répertoire approprié où stocker ses fichiers temporaires. Il est important que cet emplacement soit toujours accessible lors de l'exploitation de *LARP*, sinon le bon fonctionnement du logiciel peut en être affecté.

Si vous désirez modifier le répertoire où sont stockés les fichiers temporaires, décochez l'option **Laisser LARP déterminer le répertoire** et sélectionnez le répertoire visé via le bouton ... :

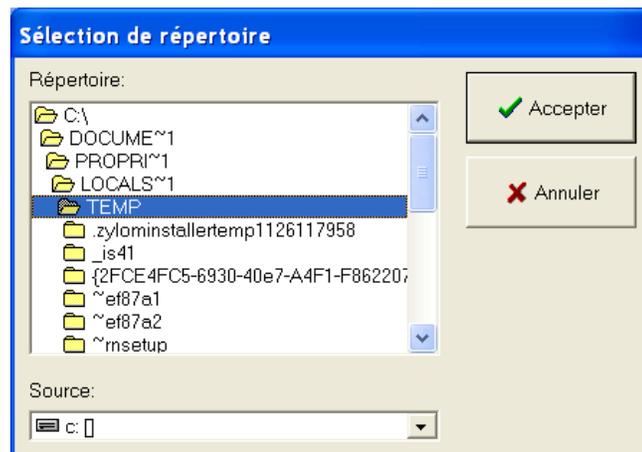


Figure 2-38 : Sélection du répertoire pour fichier temporaires

2.6.1.3 Configuration du mode super-utilisateur et du système de mises à jour

Cette section du guide réfère exclusivement à la version partagiciel de LARP.

Les options de configuration sous la rubrique **Gestion** (Figure 2-39) permettent de configurer la détection de [clés de débridage](#) activant le mode super-utilisateur et configurer le [système de mise à jour](#) intégré à *LARP*.

Le mode super-utilisateur est uniquement accessible dans la [version partagiciel](#) de *LARP* lorsque les fonctionnalités de [prévention du plagiat](#) sont activées lors de l'[installation](#) du logiciel. Lorsque les fonctionnalités de prévention du plagiat ne sont pas activées, la section **Super-utilisateur** n'est pas affichée sous l'onglet **Gestion** de la fenêtre de configuration générale de *LARP*.



Figure 2-39 : Configuration du mode super-utilisateur et du système de mise à jour

Les clés de débridage permettent de désactiver les **fonctions de prévention du plagiat** en activant le **mode super-utilisateur**. La liste déroulante énumère les gestionnaires de clés disponibles sur l'ordinateur. Lorsqu'un gestionnaire est sélectionné, le bouton **Vérier** permet de tester la présence et le bon fonctionnement d'une clé de débridage en fonction du **pseudonyme** spécifié :

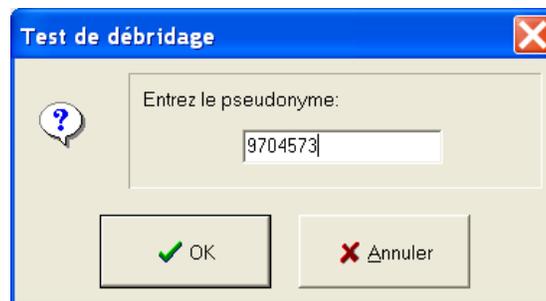


Figure 2-40 : Test de débridage

Le système de mise à jour intégré à LARP est configurable en fonction :

1. de la source des mises à jour, et
2. du moment de téléchargement de ces mises à jour.

Par défaut, les mises à jour sont téléchargées à partir d'un serveur Web géré par le fournisseur de LARP. L'adresse URL par défaut de ce serveur et du fichier répertoriant les plus récentes mises à jour est <http://larp.marcolavoie.ca/fr/Updates/updates.inf>. Si aucune adresse n'est spécifiée dans la boîte intitulée **Source** de la fenêtre de configuration (Figure 2-39), les mises à jour sont obtenues à partir du serveur par défaut. En spécifiant une adresse alternative dans cette boîte, les mises à jour seront téléchargées à partir de cette adresse. Dans la plupart des circonstances le serveur par défaut peut être exploité.

En activant l'option **Rechercher les mises à jour au démarrage de LARP**, le serveur est automatiquement et silencieusement interrogé à chaque démarrage de LARP afin de vérifier la disponibilité de nouvelles mises à jour. Si c'est le cas, LARP en informe l'utilisateur et demande une confirmation avant de les télécharger et les installer. Lorsque cette option est désactivée, les mises à jour doivent être explicitement téléchargées par l'utilisateur via la **barre de menu**. Il est fortement recommandé d'activer cette option afin de maintenir l'installation LARP à jour.

Pour plus d'information sur les clés de débridage et sur les fonctions de prévention du plagiat disponibles dans *LARP*, consultez la section traitant de la [prévention du plagiat](#). Pour de l'information supplémentaire sur la gestion des mises à jour dans *LARP*, consultez la section [Mises à jour de LARP](#).

2.6.2 Sélection de couleurs

L'environnement de développement de *LARP* peut être configuré au niveau des couleurs d'affichage dans la [console d'exécution](#) ainsi que dans les [éditeurs](#), incluant celles liées à l'[exécution pas-à-pas](#). Les options de couleurs sont accessibles via la [barre de menu](#), sous l'item **Options » Couleurs**.

La fenêtre de *configuration des couleurs* (Figure 2-41) est constituée d'onglets permettant de sélectionner les composants de l'environnement de développement à configurer : la **Console**, les **Éditeurs** et les éléments d'exécution **Pas-à-pas** à même les éditeurs. Les éléments configurables sont énumérés sous forme de boutons radio permettant de contrôler l'attribution des couleurs. La sélection des couleurs est réalisée via les contrôles de sélection situés dans la partie droite de la fenêtre de configuration.

Le bouton **Par défaut** permet de réinitialiser les couleurs aux configurations par défaut, c'est-à-dire celles présélectionnées à l'installation de *LARP*.

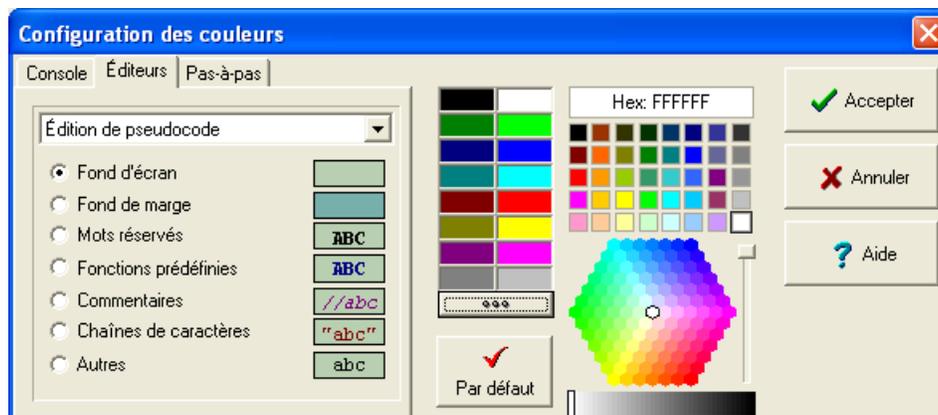


Figure 2-41 : Configuration des couleurs

2.6.2.1 Couleurs dans la console d'exécution

Les couleurs d'affichage dans la [console d'exécution](#) peuvent être configurées en fonction du type de valeur affichée par l'algorithme. Les couleurs d'affichage des trois types suivants sont configurables :

1. les [nombres entiers](#),
2. les [nombres flottants](#) et
3. les [chaînes de caractères](#).

L'affichage de ces valeurs en couleurs distinctes permet à l'utilisateur de déterminer le type d'une valeur affichée dans la console d'exécution en fonction de sa couleur.

Par défaut, les trois types de valeurs sont affichés en blanc dans la console d'exécution.

2.6.2.2 Couleurs dans les éditeurs

Pour les **éditeurs**, les attributs d'affichage suivants sont configurables du point de vue de leur couleur :

- Le fond du panneau d'édition lorsqu'un **module** pseudo-code ou organigramme est édité.
- Le fond du panneau d'édition lorsqu'un **tampon d'entrées/sorties** est édité.
- Le fond de la marge de numérotation (à gauche du panneau d'édition).

De plus, puisque l'**éditeur textuel** dispose d'une fonctionnalité de **surligne de mots réservés**, les couleurs de surligne suivantes sont aussi configurables :

- Les mots réservés du langage *LARP*.
- Le nom de **fonctions prédéfinies** du langage *LARP*.
- Les lignes de **commentaires**.
- Les **chaînes de caractères**.

La couleur d'affichage du texte non surligné dans les **modules** et les **tampons d'entrées/sorties** est aussi configurable (via l'option **Autres**). Notez que la surligne de mots réservés n'est pas appliquée au contenu des **instructions d'organigrammes**.

2.6.2.3 Couleurs pour exécution pas-à-pas

Les attributs d'affichage dans les **éditeurs** et relatifs à l'**exécution pas-à-pas** sont configurables du point de vue de leur couleur. L'**éditeur textuel** et l'**éditeur graphique** exploitent tous deux cette configuration de couleurs pour surligner les instructions de modules suivantes :

- La prochaine instruction à être exécutée en **mode d'exécution par pas et en mode marche**.
- Les instructions auxquelles sont rattachés des **points d'arrêt**.

3 Mode super-utilisateur

Cette section du guide réfère exclusivement à la [version partagiciel](#) de LARP. De plus, les fonctionnalités de prévention du plagiat doivent être activées lors de l'[installation](#) pour avoir accès au mode super-utilisateur.

Tout projet créé avec la [version partagiciel](#) de LARP dispose dès sa création de la « signature » de son auteur : le [pseudonyme](#) de l'utilisateur ayant créé le projet est irrémédiablement intégré au fichier projet. De plus, lorsque les fonctionnalités de [prévention du plagiat](#) sont activées lors de l'[installation](#) du logiciel, aucun utilisateur ne peut modifier le pseudonyme rattaché à un projet à moins d'avoir accès au **mode super-utilisateur** de LARP.

Ainsi, en mode d'utilisation normal, l'[environnement de développement](#) de LARP est bridé afin de restreindre les opportunités de plagier les projets d'utilisateurs. Le [bridage](#) de commandes LARP restreint l'accessibilité aux commandes de l'environnement de développement qui seraient susceptibles d'être exploitées par l'utilisateur pour plagier le projet d'un autre utilisateur.

L'intégration du pseudonyme de l'auteur ainsi que les autres fonctionnalités décrites dans les prochaines pages constituent les mesures de *prévention du plagiat* de LARP. Ces mesures visent à empêcher tout utilisateur de s'approprier le projet d'autrui, c'est-à-dire tricher, dans un contexte de groupe ou classe d'étudiants ayant à produire un travail scolaire avec LARP. Dans un tel contexte, le mode super-utilisateur permet à l'enseignant de [désactiver les fonctionnalités de prévention du plagiat](#) à des fins pédagogiques ou de gestion de classe telles la distribution de projets ou la correction de travaux.

Afin de restreindre l'accès au mode super-utilisateur, une [clé de débridage](#) est requise pour l'activer.

3.1 Prévention du plagiat

Cette section du guide réfère exclusivement à la [version partagiciel](#) de LARP. De plus, les fonctionnalités de prévention du plagiat doivent être activées lors de l'[installation](#) pour avoir accès au mode super-utilisateur.

Afin de prévenir le plagiat, l'[environnement de développement](#) de LARP en [version partagiciel](#) peut être *bridé*, c'est-à-dire certaines commandes de l'environnement sont désactivées ou leurs fonctionnalités sont restreintes. Le [bridage](#) de fonctionnalités dans LARP vise à empêcher les utilisateurs (considérés comme des étudiants dans un contexte de classe) d'échanger ou de partager des modules d'algorithmes.

Grâce au [bridage](#) de commandes, les documents d'un projet créé par un étudiant ne peuvent pas être transférés au projet d'un autre étudiant. De plus, toute modification illégale à un fichier projet (par exemple, en exploitant un éditeur autre que ceux de LARP) est automatiquement détectée par LARP, qui refuse alors le chargement du fichier projet dans son environnement de développement. Enfin, le [chiffrement](#) est intégré à la gestion du presse-papiers (impliqué dans les commandes [copier-coller](#)), prévenant l'exploitation de ce dernier pour transférer des modules du projet d'un étudiant à celui d'un autre étudiant.

Les fonctionnalités de prévention du plagiat de LARP sont basées sur l'unicité des [pseudonymes](#) d'utilisateur. Dans un contexte de classe regroupant plusieurs étudiants, il est donc primordial qu'à chaque étudiant soit attribué un pseudonyme distinct (tel un numéro d'étudiant).

Le [bridage](#) de commandes est seulement disponible dans la [version partagiciel](#) de LARP lorsque les fonctionnalités de prévention du plagiat sont activées lors de l'[installation](#) du logiciel. Les fonctionnalités de prévention du plagiat ne sont pas disponibles dans la [version gratuitiel](#) de LARP.

3.1.1 Pseudonyme actif

Cette section du guide réfère exclusivement à la [version partagiciel](#) de LARP. De plus, les fonctionnalités de prévention du plagiat doivent être activées lors de l'[installation](#) pour avoir accès au mode super-utilisateur.

Le [bridage](#) de l'environnement de développement de LARP en [version partagiciel](#) est basé sur l'utilisation de *pseudonymes* pour distinguer les utilisateurs d'un groupe. Le pseudonyme est une chaîne de caractères distincte identifiant l'utilisateur de LARP.

Lors du démarrage de LARP, une fenêtre de configuration (Figure 3-1) exige que l'utilisateur fournisse un pseudonyme. Si l'utilisateur ne possède aucun pseudonyme (i.e. aucun enseignant ne lui en a attribué un), il peut spécifier une chaîne de caractères (par exemple son nom ou sa date de naissance). Le pseudonyme spécifié deviendra le *pseudonyme actif* dans LARP. Lors de la création de projets LARP, le pseudonyme actif est [rattaché aux projets](#) afin d'identifier l'auteur de ces projets.

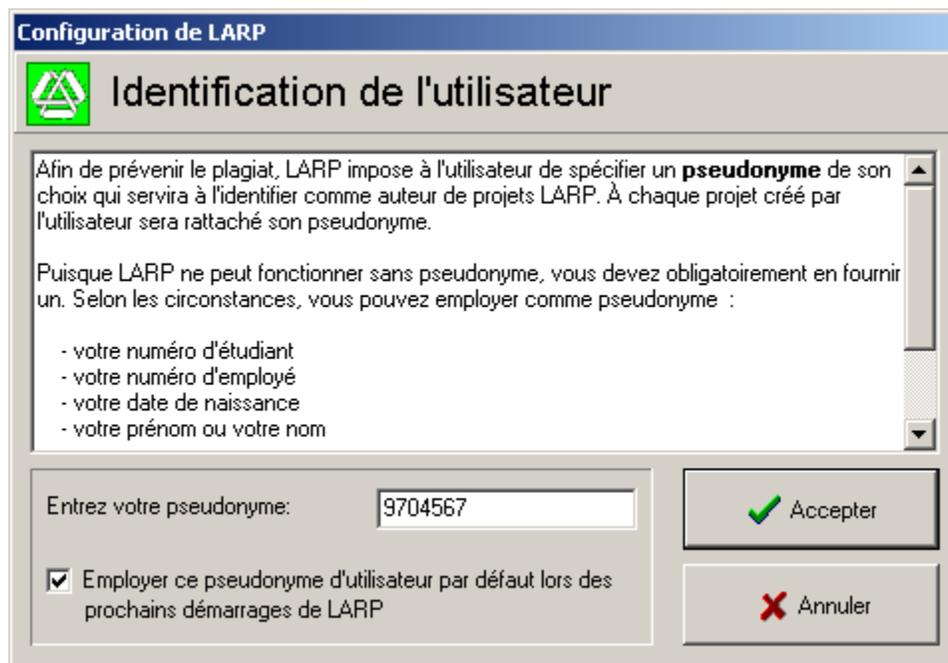


Figure 3-1 : Spécifier le pseudonyme actif

LARP offre la possibilité de toujours utiliser le même pseudonyme actif lors des démarrages subséquents. Si la case intitulée **Employer ce pseudonyme...** est cochée, le pseudonyme spécifié sera automatiquement sélectionné comme pseudonyme actif lors des prochains démarrages de LARP. La fenêtre ci-dessus ne sera donc pas affichée au démarrage.

L'utilisateur peut en tout temps modifier le pseudonyme actif via la commande **Options » Identification...** de la [barre de menu](#). Il est à noter que le projet en cours d'édition doit être fermé avant de changer le pseudonyme actif de LARP, puisque ce projet peut ne pas être édité dans le contexte du nouveau pseudonyme.

Si une [clé de débridage](#) est activée et que le pseudonyme spécifié correspond à celui de cette clé, le [mode super-utilisateur](#) est alors activé. Pour configurer le mode super-utilisateur, consultez la section [Configuration du mode super-utilisateur](#).

Le pseudonyme actif est toujours affiché dans le [panneau de statut](#). Notez cependant que l'exploitation de pseudonymes est seulement disponible dans la version partagiciel de LARP lorsque les fonctionnalités de prévention du plagiat sont activées lors de l'[installation](#) du logiciel.

Les fonctionnalités de prévention du plagiat ne sont pas disponibles dans la [version gratuit](#) de LARP.

3.1.2 Pseudonyme rattaché aux fichiers projet

Cette section du guide réfère exclusivement à la [version partagiciel](#) de LARP. De plus, les fonctionnalités de prévention du plagiat doivent être activées lors de l'[installation](#) pour avoir accès au mode super-utilisateur.

Lors de la création d'un nouveau projet LARP (via la [barre de menu](#) ou le [panneau de contrôle](#)), le [pseudonyme actif](#) au moment de la création (i.e. celui apparaissant dans le [panneau de statut](#)) est rattaché au projet de façon permanente. En aucun temps l'étudiant (i.e. l'utilisateur) peut ultérieurement modifier le pseudonyme rattaché au projet. Lorsqu'un projet est sauvegardé dans un fichier projet, le pseudonyme lui étant rattaché est aussi sauvegardé dans ce même fichier, accompagnant ainsi le projet.

Lors du chargement d'un fichier projet existant, LARP n'autorise le chargement que si le pseudonyme rattaché au projet (lu à même le fichier) est identique à celui actif (apparaissant dans le panneau de statut). Ainsi, un étudiant est dans l'impossibilité de charger dans l'éditeur de LARP un fichier projet créé par un autre étudiant, à moins d'usurper son pseudonyme. Cependant, même en usurpant l'identité de l'étudiant auteur du fichier projet chargé, l'utilisateur illégitime est dans l'impossibilité de changer le pseudonyme rattaché au projet, et ainsi de remettre à l'enseignant le projet usurpé comme étant le sien.

Cette fonctionnalité assure ainsi à l'enseignant qu'un seul étudiant puisse soumettre un même fichier projet, puisqu'un seul pseudonyme est rattaché au fichier projet et que ce pseudonyme ne peut être modifié.

En utilisant le mode super-utilisateur, l'enseignant peut [désactiver cette fonctionnalité](#) et charger dans LARP le projet de tout étudiant, sans égard au pseudonyme actif.

Lors de la création d'un projet en mode super-utilisateur, aucun pseudonyme n'est rattaché au nouveau projet (i.e. le projet n'a pas d'auteur). On peut considérer un projet sans pseudonyme rattaché comme un [projet public](#), pouvant être chargé par quiconque et sans égard au pseudonyme actif. Lorsqu'un étudiant charge dans LARP un projet public, le pseudonyme actif est automatiquement rattaché au projet chargé. Ainsi, lorsque l'étudiant aura modifié et sauvegardé le projet, son pseudonyme sera dorénavant rattaché au fichier projet et aucun autre étudiant ne pourra charger ce projet dans LARP afin d'en plagier le contenu.

Notez que l'exploitation de pseudonymes est seulement disponible dans la [version partagiciel](#) de LARP lorsque les fonctionnalités de prévention du plagiat sont activées lors de l'[installation](#) du logiciel. Les fonctionnalités de prévention du plagiat ne sont pas disponibles dans la [version gratuit](#) de LARP. Notez de plus que les fichiers projets ne sont pas transférables entre la version partagiciel et la version gratuit de LARP, afin d'éviter que des utilisateurs exploitent la version gratuit (n'exploitant pas les pseudonymes) pour échanger des fichiers projet.

3.1.3 Chiffrement des documents

Cette section du guide réfère exclusivement à la [version partagiciel](#) de LARP. De plus, les fonctionnalités de prévention du plagiat doivent être activées lors de l'[installation](#) pour avoir accès au mode super-utilisateur.

Afin d'empêcher un utilisateur d'accéder au contenu du fichier projet d'un autre utilisateur pour en extraire les modules, la [version partagiciel](#) de LARP exploite le chiffrement numérique pour brouiller le contenu des documents. Lorsque LARP transcrit un projet dans son fichier, le contenu des documents du projet (i.e. ses modules et ses tampons d'entrées/sorties) est chiffré en utilisant le [pseudonyme actif](#) comme clé de chiffrement (un algorithme de chiffrement robuste et public, *Blowfish*, est exploité par LARP pour le chiffrement de documents).

Le contenu d'un fichier projet *LARP* étant chiffré, un utilisateur illégitime est dans l'impossibilité d'utiliser un éditeur autre que ceux de *LARP* pour accéder aux documents contenus dans le fichier. En d'autres mots, le chiffrement des fichiers projet *LARP* à l'aide du pseudonyme de l'auteur assure la confidentialité du projet, seul l'auteur étant apte à déchiffrer le contenu de ses fichiers projet.

De plus, afin d'empêcher un utilisateur illégitime d'altérer illégalement un fichier projet *LARP* via un éditeur de texte autre que celui de *LARP*, un *code de vérification d'intégrité* est inséré dans chaque fichier projet *LARP*. Si le fichier projet est altéré illégalement, la modification sera automatiquement détectée par *LARP* lors du prochain chargement du fichier projet dans l'[environnement de développement](#).

La seule façon pour un utilisateur illégitime de charger un projet *LARP* créé par un autre utilisateur est d'usurper son identité (i.e. spécifier le pseudonyme de l'auteur du projet comme pseudonyme actif). Cependant, puisque l'utilisateur illégitime est par la suite dans l'impossibilité de changer le pseudonyme rattaché au projet (voir la section traitant des [pseudonymes rattachés aux fichiers projet](#)), il ne peut en aucun temps présenter le projet usurpé comme étant sien.

En utilisant le [mode super-utilisateur](#), un enseignant peut [désactiver cette fonctionnalité](#) de prévention du plagiat et charger dans l'environnement de développement de *LARP* les projets de ses étudiants.

Notez que l'exploitation de pseudonymes pour chiffrer les fichiers projet est seulement disponible dans la version partagiciel de *LARP* lorsque les fonctionnalités de prévention du plagiat sont activées lors de l'[installation](#) du logiciel. Les fonctionnalités de prévention du plagiat ne sont pas disponibles dans la [version gratuiciel](#) de *LARP*.

3.1.4 Contrôle du copier-coller

*Cette section du guide réfère exclusivement à la [version partagiciel](#) de *LARP*. De plus, les fonctionnalités de prévention du plagiat doivent être activées lors de l'[installation](#) pour avoir accès au mode super-utilisateur.*

Afin d'empêcher quiconque d'exploiter le presse-papiers pour transférer des modules d'un projet *LARP* en [version partagiciel](#) au projet d'un autre utilisateur, le contenu du presse-papiers est [chiffré](#) à l'aide du [pseudonyme actif](#) apparaissant dans le [panneau de statut](#). Ainsi, les commandes de l'[environnement de développement](#) impliquant le presse-papiers (couper, copier et coller) restreignent l'utilisateur à copier du pseudo-code et des éléments d'organigrammes d'un module vers un autre module du même projet, ou vers un module d'autres projets du même auteur. Grâce au chiffrement, la version partagiciel de *LARP* ne permet pas à l'utilisateur de coller le contenu du presse-papiers au projet d'un autre utilisateur ou dans une application autre que *LARP*.

Le chiffrement de documents est uniquement appliqué au presse-papiers lorsque le contenu y étant copié provient d'un module. Lorsque la source est un tampon d'entrées/sorties, le contenu n'est pas chiffré et peut ainsi être collé dans un tampon d'entrées/sorties de tout autre projet, ou dans un éditeur autre que celui de *LARP*. Par définition, les tampons d'entrées/sorties contiennent des données ou des résultats, qui sont généralement considérés comme publiques.

En exploitant le mode super-utilisateur, un enseignant peut [désactiver cette fonctionnalité](#) de prévention du plagiat et utiliser le presse-papiers pour copier le contenu de modules dans n'importe quel projet édité avec *LARP* ou avec tout autre éditeur.

Notez que l'exploitation de pseudonymes pour chiffrer le contenu du presse-papiers est seulement disponible dans la version partagiciel de *LARP* lorsque les fonctionnalités de prévention du plagiat sont activées lors de l'[installation](#) du logiciel. Les fonctionnalités de prévention du plagiat ne sont pas disponibles dans la [version gratuiciel](#) de *LARP*.

3.1.5 Contrôle de l'impression

Cette section du guide réfère exclusivement à la [version partagiciel](#) de LARP. De plus, les fonctionnalités de prévention du plagiat doivent être activées lors de l'[installation](#) pour avoir accès au mode super-utilisateur.

Afin d'éviter qu'un étudiant produise une copie papier d'un projet LARP pour le distribuer à ses pairs, la commande d'impression de [modules](#) est bridée dans la [version partagiciel](#) de LARP, sauf en [mode super-utilisateur](#). Tout utilisateur peut cependant imprimer les [tampons d'entrées/sorties](#).

Lorsque la commande d'impression est invoquée, une fenêtre de sélection de document est affichée :

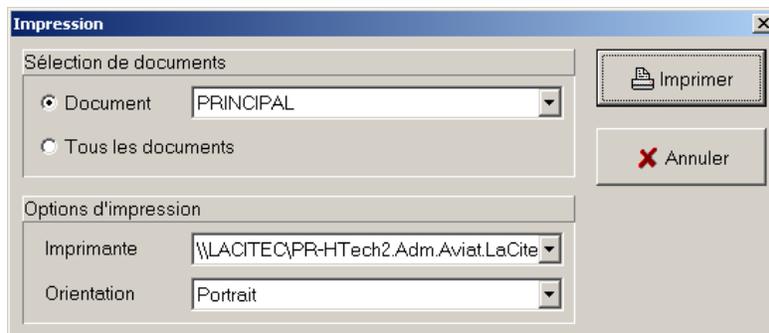


Figure 3-2 : Impression des documents

L'utilisateur peut ainsi sélectionner le module ou tampon d'entrées/sorties du projet à imprimer, ou imprimer l'ensemble du projet. Les options d'impression incluent la sélection d'imprimante et l'orientation du papier d'impression. Lorsque le mode super-utilisateur est désactivé, seuls les tampons d'entrées/sorties peuvent être imprimés.

Notez que le contrôle de l'impression est seulement disponible dans la version partagiciel de LARP lorsque les fonctionnalités de prévention du plagiat sont activées lors de l'[installation](#) du logiciel. L'impression de modules est accessible en tout temps dans la [version gratuitiel](#) de LARP ainsi que dans la version partagiciel lorsque les fonctionnalités de prévention du plagiat ne sont pas activées.

3.2 Débridage de l'environnement de développement

Cette section du guide réfère exclusivement à la [version partagiciel](#) de LARP. De plus, les fonctionnalités de prévention du plagiat doivent être activées lors de l'[installation](#) pour avoir accès au mode super-utilisateur.

Dans un contexte de classe d'étudiants ayant à produire des algorithmes avec la [version partagiciel](#) de LARP dans le cadre d'évaluations scolaires, l'enseignant doit avoir la possibilité de désactiver les fonctionnalités de prévention du plagiat afin de créer des fichiers projet LARP à être distribués aux étudiants, ainsi qu'à charger dans LARP les projets de ces derniers pour fins d'évaluation. Le mode *super-utilisateur* permet à l'enseignant d'effectuer ces tâches en désactivant les fonctionnalités de prévention du plagiat.

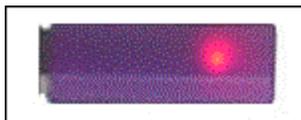


Figure 3-3 : Clé de débridage (port USB)

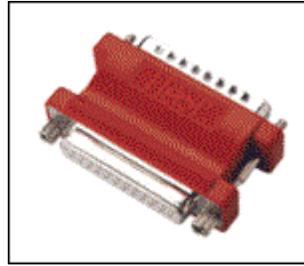


Figure 3-4 : Clé de débridage (port parallèle)

Afin d'interdire aux étudiants l'accès au mode super-utilisateur, une *clé de débridage* est requise pour activer le mode super-utilisateur. Les clés de débridage pour *LARP* sont des périphériques physiques à brancher dans le port USB (*Universal Serial Bus*) ou le port parallèle de l'ordinateur. Chaque clé de débridage est vendue pré-configurée avec un pseudonyme exclusif.

À chaque démarrage de la version partagiciel de *LARP*, le logiciel interroge les ports USB et parallèles de l'ordinateur afin de détecter la présence d'une clé de débridage. Si une telle clé est détectée et que son pseudonyme correspond à celui de l'utilisateur (i.e. le **pseudonyme actif**), le mode super-utilisateur est automatiquement activé. L'activation du mode super-utilisateur est indiquée dans le **panneau de statut**, où le pseudonyme actif est remplacé par les lettres **SU** (pour Super-Utilisateur) dans le champ correspondant :

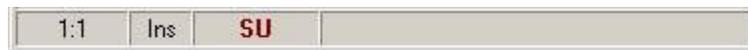


Figure 3-5 : Indicateur du mode super-utilisateur

Pour commander des clés de débridage *LARP*, consultez la section [Commander des clés de débridage](#).

Notes : la présence de clé de débridage est uniquement vérifiée au démarrage de la version partagiciel de *LARP* et lors d'un changement de pseudonyme via la [barre de menu](#). En tout autre temps la clé de débridage peut être débranchée sans pour autant désactiver le mode super-utilisateur.

Notez aussi que les clés de débridage peuvent seulement être exploitées dans la version partagiciel de *LARP* lorsque les fonctionnalités de prévention du plagiat sont activées à l'[installation](#) du logiciel. Le bridage de fonctionnalités n'est pas disponible dans la [version gratuit](#) de *LARP* ainsi que dans la version partagiciel lorsque les fonctionnalités de prévention du plagiat ne sont pas activées.

3.2.1 Sélection d'une technologie de clé

*Cette section du guide réfère exclusivement à la [version partagiciel](#) de *LARP*. De plus, les fonctionnalités de prévention du plagiat doivent être activées lors de l'[installation](#) pour avoir accès au mode super-utilisateur.*

Puisque plusieurs technologies de [clés de débridage](#) sont acceptées par *LARP*, l'utilisateur doit configurer le logiciel en spécifiant la technologie correspond à la clé de débridage qu'il entend employer. Consultez la section [Configuration du mode super-utilisateur](#) pour plus d'information sur la sélection d'une technologie de clé.

3.2.2 Statistiques de projet

Lorsqu'un utilisateur crée et développe un projet *LARP*, les statistiques suivantes sont accumulées pour le projet :

- la date de création du projet;
- le temps total consacré par l'utilisateur à [éditer les documents](#) du projet dans *LARP*;
- le nombre de [compilations](#) du projet effectuées depuis sa création; et
- le nombre de sauvegardes dans le fichier projet (excluant les sauvegardes de sécurité) effectuées par l'utilisateur depuis la création du projet.

Ces statistiques permettent à l'enseignant d'avoir un aperçu des heures consacrées par l'étudiant à son projet. Des statistiques improbables peuvent ainsi mettre en évidence le plagiat. Par exemple, un projet constitué de plusieurs documents mais n'ayant été édité que quelques minutes est généralement considéré suspect.

Dans la [version partagiciel](#) de *LARP* avec les fonctionnalités de [prévention du plagiat](#) activées, les statistiques accumulées sur un projet *LARP* sont exclusivement accessibles en [mode super-utilisateur](#), via la [barre de menu](#). Dans la version partagiciel sans prévention de plagiat ainsi que dans la [version gratuiciel](#) de *LARP*, les statistiques sont toujours accessibles, aucun pseudonyme n'étant cependant affiché.

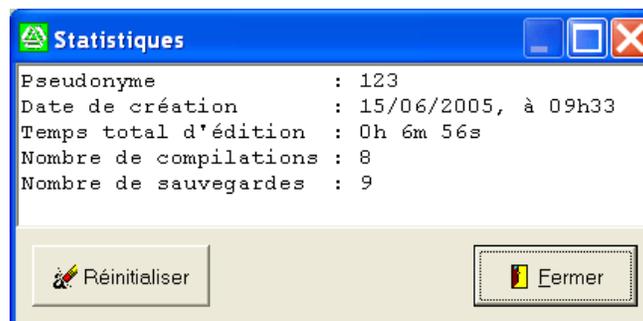


Figure 3-6 : Statistiques de projet

Le bouton **Réinitialiser** permet de remettre à zéro les statistiques associées au projet édité, incluant le [pseudonyme rattaché](#), qui est alors effacé. Ainsi, l'enseignant peut rendre « [publique](#) » le projet *LARP* d'un étudiant.

3.2.3 Conversion d'organigramme en pseudo-code

LARP permet de convertir tout module sous forme d'organigramme en module pseudo-code. Lorsqu'un organigramme est affiché dans l'[éditeur graphique](#), la commande de la [barre de menu](#) **Afficher » Pseudo-code...** affiche une fenêtre contenant le pseudo-code équivalent à l'organigramme affiché :

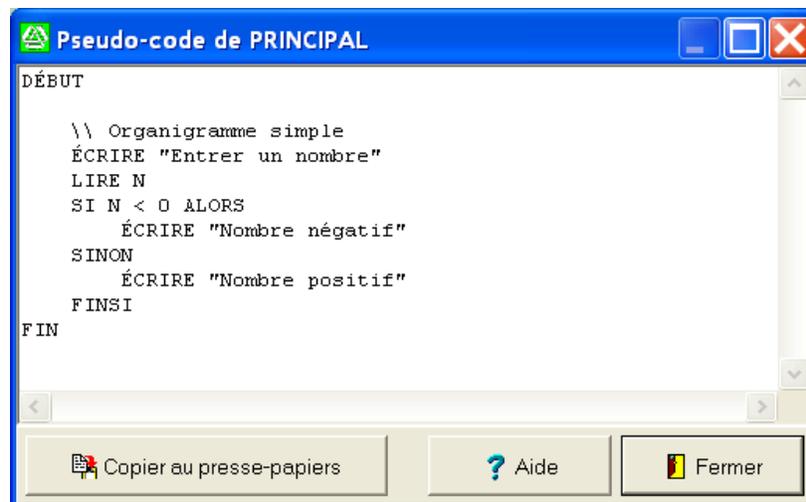


Figure 3-7 : Conversion d'un organigramme en pseudo-code

Le bouton **Copier au presse-papiers** permet de copier le pseudo-code sous forme textuelle le pseudo-code de la fenêtre vers le presse-papiers. Ce pseudo-code peut ainsi être réintroduit dans un module du projet par [copier-coller](#).

Dans la [version partagiciel](#) de LARP avec les fonctionnalités de [prévention du plagiat](#) activées, cette fonctionnalité de conversion est limitée au [mode super-utilisateur](#) afin de permettre à un enseignant d'imposer aux élèves qu'ils écrivent leurs algorithmes en pseudo-code. Dans la version partagiciel sans prévention de plagiat ainsi que dans la [version gratuitiel](#) de LARP, la fonctionnalité de conversion est toujours disponible.

3.2.4 Projets publics

Cette section du guide réfère exclusivement à la [version partagiciel](#) de LARP. De plus, les fonctionnalités de [prévention du plagiat](#) doivent être activées lors de l'[installation](#) pour avoir accès au mode super-utilisateur.

Lorsqu'un projet est créé par un utilisateur avec la [version partagiciel](#) de LARP avec fonctionnalités de [prévention du plagiat](#) activées, son pseudonyme (i.e. le [pseudonyme actif](#)) est rattaché au projet de façon permanente, et seul cet utilisateur peut ultérieurement charger dans LARP le fichier projet (voir la section [Pseudonyme rattaché aux fichiers projet](#)).

Dans un tel contexte, comment l'enseignant peut-il distribuer à des étudiants un fichier projet LARP à être complété par ces derniers ?

En fait, lorsqu'un projet est créé en [mode super-utilisateur](#), aucun pseudonyme n'est rattaché au projet lorsque ce dernier est sauvegardé dans son fichier. Lorsque l'utilisateur tente de charger un fichier projet n'ayant aucun pseudonyme rattaché, LARP permet le chargement du projet quel que soit le pseudonyme actif (i.e. tous les utilisateurs peuvent charger le projet), puis rattache le pseudonyme actif au projet chargé.

En résumé, un enseignant désirant distribuer un fichier projet doit créer le projet en mode super-utilisateur et le sauvegarder ainsi, créant ainsi un *fichier projet public*. Tout étudiant peut ensuite charger ce fichier projet dans LARP et le modifier à sa guise. Lorsqu'il sauvegarde le projet modifié, son pseudonyme est automatiquement rattaché au projet dans le fichier modifié, prévenant ainsi toute tentative ultérieure de plagiat.

Un super-utilisateur peut en tout temps transformer un projet avec pseudonyme rattaché en projet public (i.e. sans pseudonyme rattaché) via la réinitialisation des [statistiques du projet](#).

Notez que les fichiers projets créés avec la version partagiciel de *LARP* sans fonctionnalités de prévention du plagiat ainsi que ceux créés avec la [version gratuitiel](#) de *LARP* sont toujours publics puisque aucun pseudonyme n'est rattaché à ces fichiers. Notez de plus que les fichiers projets ne sont pas transférables entre la version partagiciel et la version gratuitiel de *LARP*, afin d'éviter que des utilisateurs exploitent la version gratuitiel (n'exploitant pas les pseudonymes) pour échanger des fichiers projet.

4 Un premier algorithme

La syntaxe du langage pseudo-code de *LARP* est souple et intuitive, comme vous pourrez le constater dans la suite de ce document.

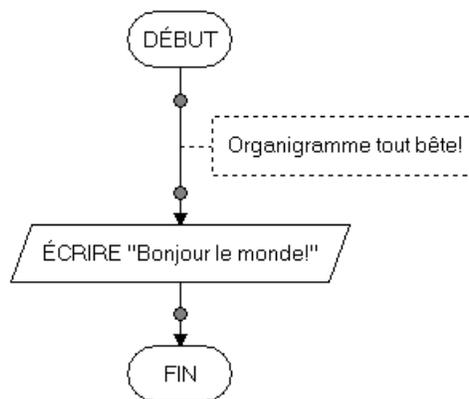
Pour illustrer cette syntaxe, voici un premier algorithme qui ne fait qu'afficher la chaîne de caractères **Bonjour le monde!** à l'écran. L'algorithme est formulé sous forme de pseudo-code et d'organigramme :

```

\\ Pseudo-code tout bête!
DÉBUT
  ÉCRIRE "Bonjour le monde!"
FIN

```

Pseudo-code 4-1 : Pseudo-code tout bête



Organigramme 4-1 : Organigramme tout bête

Les différents composants de cet algorithme sont expliqués dans les prochaines sections.

4.1 Les commentaires

Comme dans la plupart des langages de programmation, on peut insérer des commentaires dans les algorithmes de *LARP*.

Dans le pseudo-code *LARP*, il faut précéder une ligne de commentaires des symboles `\\` (deux *antéslashes* consécutifs) : tout ce qui suit jusqu'à la fin de la ligne est alors ignoré lors de la [compilation](#) de l'algorithme. Pour prolonger un commentaire sur plusieurs lignes, il faut commencer chaque ligne par `\\` :

```

\\ Voici un exemple de commentaire se
\\ propageant sur plus d'une ligne
DÉBUT
  ÉCRIRE "Bonjour le monde!"      \\ Commentaire en fin de ligne
FIN

```

Pseudo-code 4-2 : Les commentaires

Dans les organigrammes *LARP*, l'instruction *Commentaire* (Figure 4-1), disponible via le [panneau de modèles](#) ou via le [menu contextuel de l'éditeur graphique](#), permet d'insérer des commentaires dans les organigrammes (tel que dans l'Organigramme 4-1).

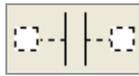


Figure 4-1 : Modèles de commentaire

4.2 Début et fin d'un algorithme

Un algorithme *LARP* doit débuter par l'instruction **DÉBUT**. Cette instruction indique le point de départ de l'exécution de l'algorithme. L'instruction suivant **DÉBUT** (dans le Pseudo-code 4-1 et l'Organigramme 4-1, **ÉCRIRE "Bonjour le monde!"**) est la première instruction exécutée.

Réciproquement, l'instruction **FIN** indique le point où l'algorithme se termine. C'est à ce point que l'exécution de l'algorithme cesse.

Comme nous le verrons ultérieurement, un algorithme *LARP* peut être divisé en [modules](#), où chaque module est un pseudo-code ou un organigramme distinct. Dans un tel contexte, un seul de ces modules doit contenir les instructions **DÉBUT** et **FIN**. Ce sont ces instructions qui indiquent où débute et cesse l'exécution de l'algorithme parmi les modules le constituant.

Puisqu'un algorithme contient généralement un seul point de départ de son exécution, un seul module de l'algorithme doit débiter par l'instruction **DÉBUT**. Similairement, l'algorithme doit contenir une seule instruction **FIN**, et celle-ci doit être la dernière instruction du module comprenant l'instruction **DÉBUT**. Ce module est appelé le [module principal](#). Si l'algorithme comprend d'autres modules, ceux-ci sont dits [modules auxiliaires](#).

Lorsqu'un [nouveau projet](#) est créé dans l'environnement de développement de *LARP*, les instructions **DÉBUT** et **FIN** sont automatiquement insérées dans le module principal du projet.

4.3 Syntaxe des instructions

Comme langage de programmation, *LARP* offre un ensemble d'instructions permettant de formuler des algorithmes sous forme de pseudo-code et d'organigramme. Ces instructions acceptent optionnellement un ou plusieurs arguments et leur syntaxe correspond à l'un des deux formats suivants :

instruction arg1, arg2, ...

ou

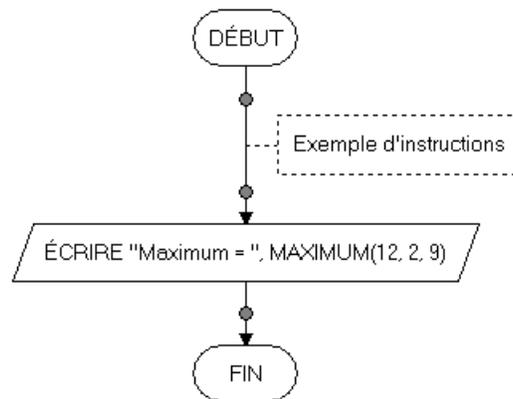
instruction (arg1, arg2, ...)

L'exemple ci-dessous (voir Pseudo-code 4-3 et Organigramme 4-2) utilise l'instruction **ÉCRIRE** afin d'afficher un résultat dans la console d'exécution. Une même instruction **ÉCRIRE** peut afficher plus d'un résultat :

```
\\ Exemple d'instructions
DÉBUT
  ÉCRIRE "Maximum = ", MAXIMUM(12, 2, 9)
FIN
```

Pseudo-code 4-3 : Syntaxe des instructions

Comme on le constate dans cet exemple, l'instruction **ÉCRIRE** adopte la première forme de syntaxe alors que l'instruction **MAXIMUM** adopte la deuxième forme de syntaxe.



Organigramme 4-2 : Syntaxe des instructions

Notez que les instructions du langage *LARP* peuvent être écrites avec accents (**ÉCRIRE**, **JUSQU'À**, **SÉLECTION**, ...) ou sans accents (**ECRIRE**, **JUSQU'A**, **SELECTION**, ...). Puisque la grande majorité des langages de programmation n'acceptent pas les accents, *LARP* supporte de facto ce « standard ».

Notez aussi que *LARP* ne fait aucune distinction entre les lettres majuscules et les lettres minuscules. Ainsi, les instructions **ÉCRIRE**, **JUSQU'À** et **SÉLECTION** peuvent aussi bien s'écrire respectivement **Écrire**, **Jusqu'à** et **Sélection**, ou **écrire**, **jusqu'à** et **sélection**.

Dans les textes d'[aide en ligne](#) de *LARP*, toutes les instructions retrouvées dans les exemples sont formulées en lettres majuscules (avec accents). Cette pratique permet d'identifier rapidement les mots réservés du langage *LARP* dans un pseudo-code ou un organigramme.

4.4 Séparation des instructions

Les instructions pseudo-code d'un module sont généralement écrites sur des lignes séparées. Ainsi, un changement de ligne (i.e. l'insertion d'un retour de chariot à la fin d'une ligne) indique la fin d'une instruction et le début de la suivante. Cependant il est possible de prolonger une longue instruction sur la ligne suivante en terminant la première ligne de l'instruction par le symbole **\$** :

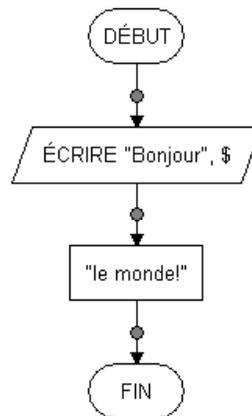
```

\\ Pseudo-code moins bête
DÉBUT
  LIRE a, b, c
  ÉCRIRE "Le maximum parmi les trois valeurs ", a, ", ", b, " et ", $
          c, " est ", MAXIMUM(a, b, c)
FIN
  
```

Pseudo-code 4-4 : Séparation des instructions

Une longue instruction peut ainsi être prolongée sur plusieurs lignes, chaque ligne sauf la dernière étant terminée par **\$**.

Le même principe s'applique aux instructions d'organigramme. Une instruction peut être terminée par le symbole \$ et ensuite poursuivie dans l'instruction séquentielle suivante :



Organigramme 4-3 : Prolongation d'une instruction d'organigramme

4.5 Création du projet *LARP*

Voici les étapes menant à la création et l'exécution d'un premier algorithme avec la [version partagiciel](#) de *LARP*. Si vous utilisez la [version gratuitel](#) de *LARP*, les étapes 2 et 3 doivent être ignorées :

1. Démarrez *LARP* et fermez la *fenêtre d'accueil* (celle-ci se ferme automatiquement après quelques secondes si vous ne le faites pas).



Figure 4-2 : Fenêtre d'accueil

2. Si le partagiciel n'est pas enregistré, la *fenêtre d'enregistrement* (Figure 4-3) est alors affichée. Cette fenêtre indique comment enregistrer votre installation *LARP*. Si c'est le cas, pressez le bouton **Enregistrer plus tard**. Pour plus d'information sur l'enregistrement de votre installation *LARP*, consultez la section [Enregistrement](#). Notez que cette fenêtre n'est pas affichée si le partagiciel est enregistré ou si vous utilisez la version gratuitel de *LARP*.



Figure 4-3 : Fenêtre d'enregistrement

3. Lorsque les fonctionnalités de **prévention du plagiat** sont activées, tout utilisateur de *LARP* doit spécifier son **pseudonyme** (Figure 4-4). Ce pseudonyme est exploité pour identifier l'auteur du projet dans un cadre éducationnel. Si vous n'avez pas de pseudonyme, entrez un nombre quelconque. Si de plus vous cochez la case **Employer ce pseudonyme...**, *LARP* utilisera le pseudonyme spécifié aux démarrages subséquents du logiciel :

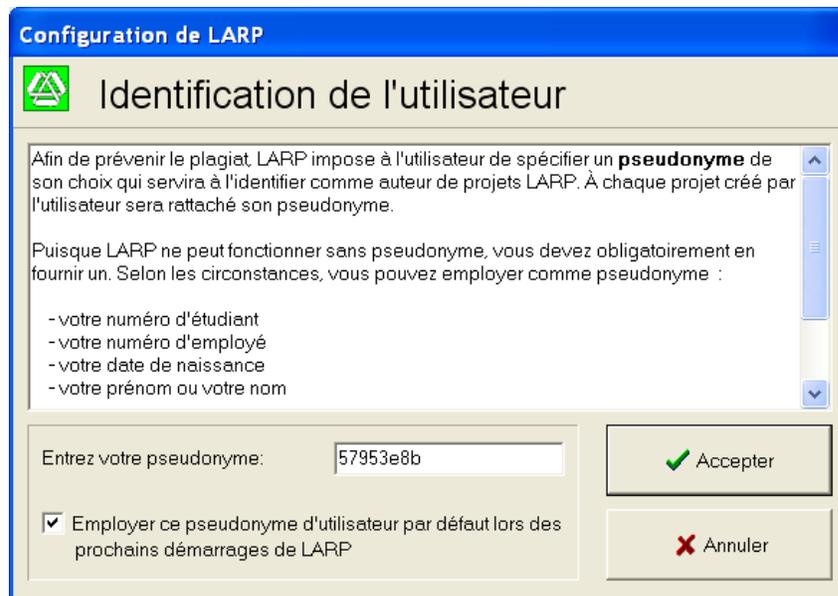


Figure 4-4 : Pseudonyme d'utilisateur

La fenêtre d'identification de l'utilisateur n'est pas affichée lorsque les fonctionnalités de prévention du plagiat ne sont pas activées, ce qui est toujours le cas pour la version gratuitel de *LARP*.

4. Si votre installation LARP est **enregistrée** et votre ordinateur relié à Internet, le **système de mise à jour** intégré tente alors de déterminer si des nouvelles mises à jour sont disponibles pour LARP. Si c'est le cas vous en êtes averti. Pour l'instant vous pouvez refuser l'installation des mises à jour; elles pourront être installées ultérieurement.

La mise à jour automatisée n'est pas disponible dans la version gratuitel de LARP.

5. Sélectionnez la commande **Fichier » Nouveau...** à partir de la **barre de menu**. Cette commande crée un nouveau projet ou document LARP. La fenêtre *Nouveau* (Figure 4-5) demande ce qui doit être créé; puisqu'il n'y a présentement aucun projet dans LARP, ce dernier ne permet pas de créer des documents (**modules** ou **tampons d'entrées/sorties**). Sélectionnez l'option **Pseudo-code** afin de créer un nouveau projet avec pseudo-code dans le module principal, puis appuyez sur le bouton **Accepter**.

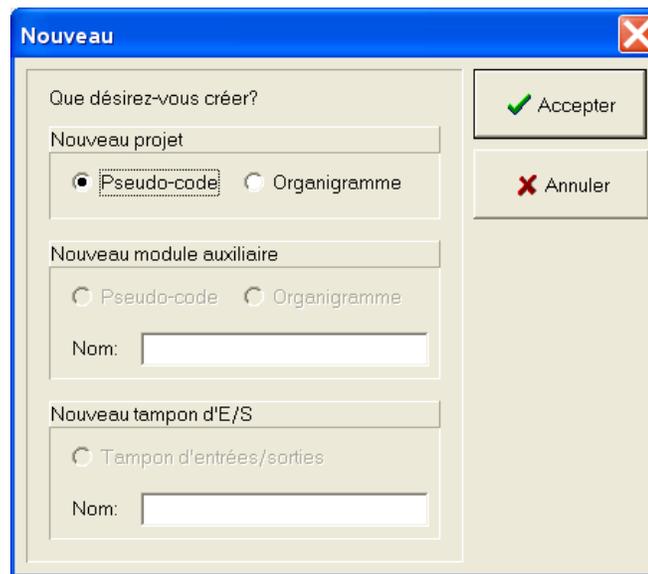


Figure 4-5 : Fenêtre Nouveau

6. L'**éditeur textuel** affiche alors un pseudo-code minimal constituant le module principal du projet et délimité par les instructions **DÉBUT** et **FIN**. Insérez l'instruction **ÉCRIRE** telle que présentée dans le pseudo-code suivant :

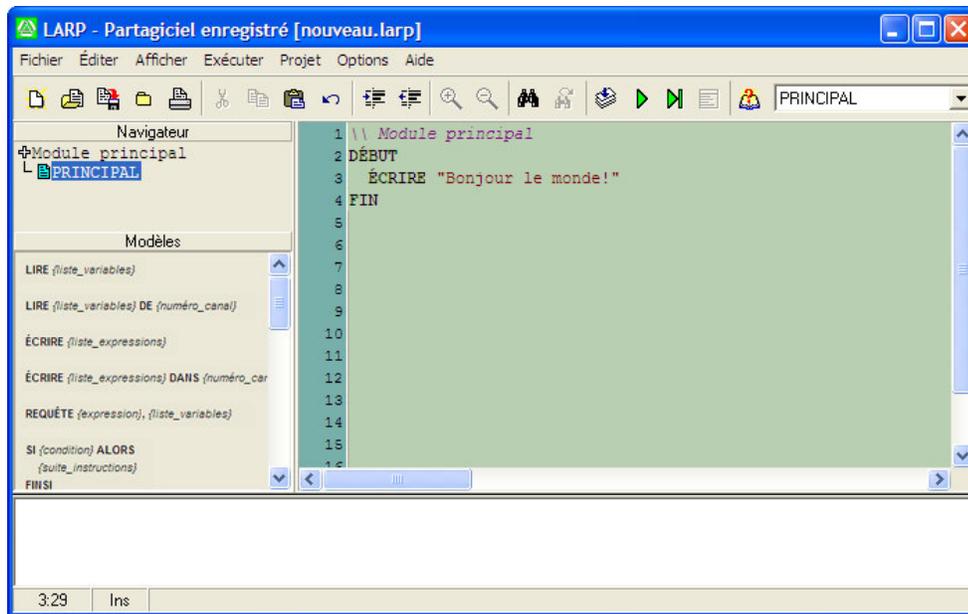


Figure 4-6 : Projet LARP en cours d'édition

7. Pour exécuter l'algorithme, sélectionnez la commande **Projet » Exécuter...** dans la **barre de menu**. Cette commande exécute le projet LARP et affiche les résultats dans la **console d'exécution** (Figure 4-7). Comme indiqué dans la console, appuyez sur une touche du clavier pour fermer celle-ci et retourner à l'environnement de développement de LARP.

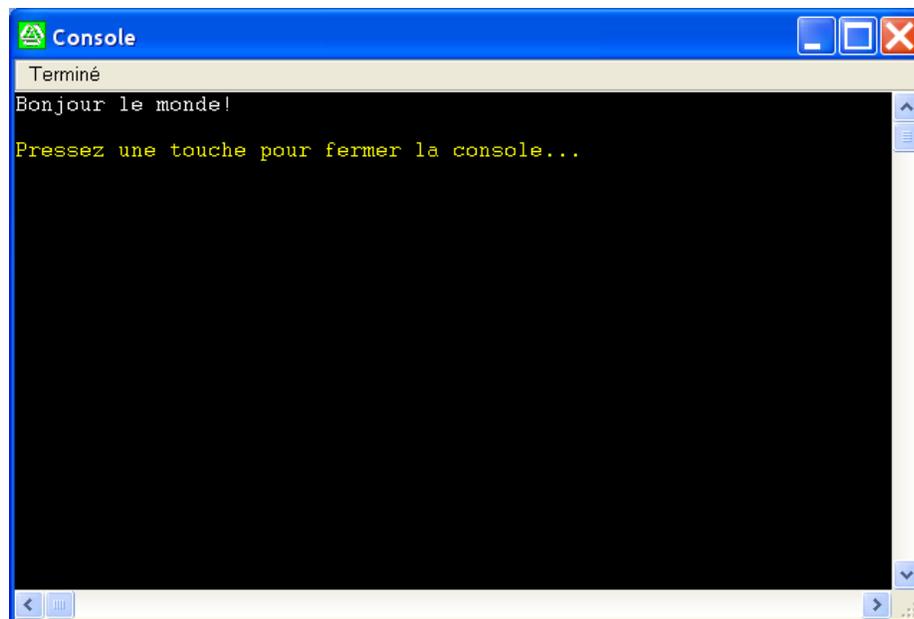


Figure 4-7 : Exécution du projet

Il se peut que la commande d'exécution échoue si vous avez entré le pseudo-code de façon erronée dans l'éditeur textuel. Dans ce cas la console d'exécution n'apparaît pas et des messages d'erreurs apparaissent dans le [panneau de messages](#) (Figure 4-8).

8. Notez que le **panneau de messages** (Figure 4-8) affiche de l'information relative à la compilation et l'exécution du projet. Si vous avez entré le pseudo-code de façon erronée dans l'éditeur textuel, les erreurs seront identifiées dans le panneau de messages.

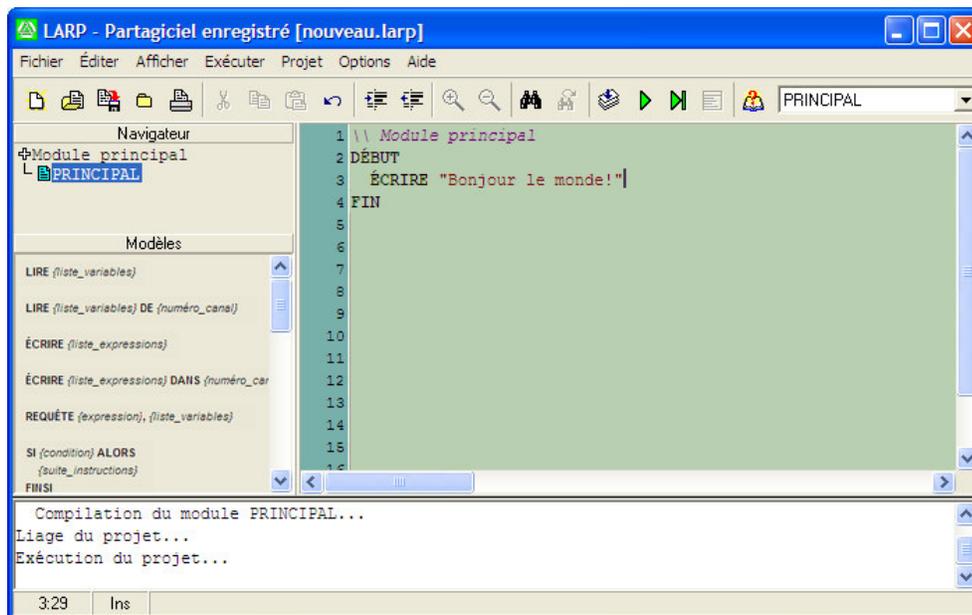


Figure 4-8 : Messages générés lors de la compilation et l'exécution

9. Si vous désirez sauvegarder votre travail dans un fichier projet, sélectionnez la commande **Fichier** » **Sauvegarder** dans la **barre de menu**. Vous devez alors spécifier le répertoire et le nom du fichier projet :

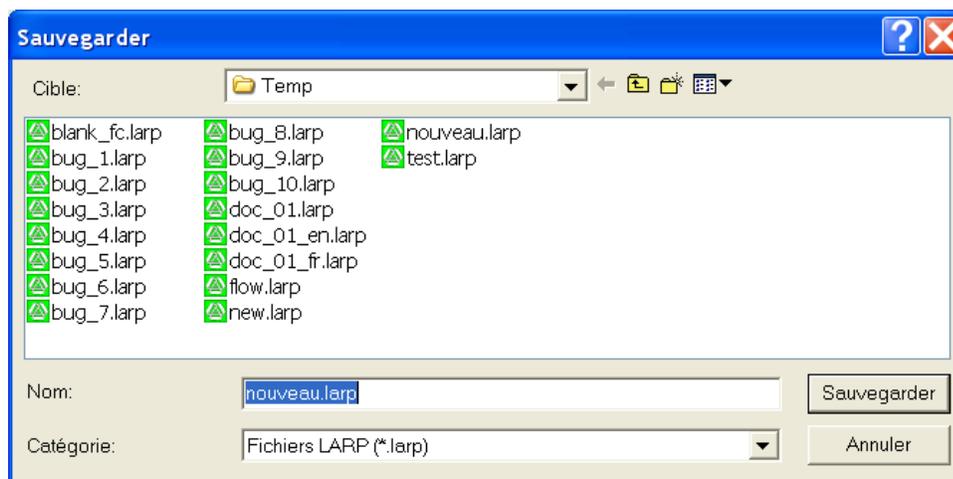


Figure 4-9 : Sauvegarder le projet

Et voilà! Vous venez de créer, exécuter et sauvegarder votre premier projet *LARP*. Maintenant vous pouvez explorer *LARP* à votre guise et développer des algorithmes complexes. Le *Guide d'utilisation de LARP* contient toute l'information requise pour produire de tels algorithmes.

5 Constantes et variables

Dans la syntaxe des pseudo-codes et organigrammes *LARP*, une *constante* est **numérique** ou **alphanumérique**. Par exemple, **12.3** est une constante numérique représentant un nombre fractionnel, et **"allo"** est une constante alphanumérique représentant une chaîne de caractères. *LARP* supporte plusieurs types de constantes.

Une *variable* dans un algorithme *LARP* (comme dans la plupart des langages de programmation tels *C++* et *Java*) est un emplacement dans la mémoire de l'ordinateur où sont stockées des données.

Une variable peut être vue comme une boîte dans laquelle sont rangées des informations qui peuvent être récupérées en tout temps. À la différence de la boîte qui redevient vide lorsqu'on en retire son contenu, la variable stocke une « copie » des données que l'algorithme y range (via l'*affectation*). Ainsi, lorsque l'algorithme récupère les données d'une variable, il récupère en fait une copie du contenu de la variable. Cette dernière conserve donc ses données (i.e. son contenu), qui peuvent être récupérées à de multiples reprises. En fait, une variable conserve ses données jusqu'à ce que d'autres données y soient stockées, remplaçant les données antérieures, ou jusqu'à ce que la variable soit détruite par l'algorithme.

Puisqu'un algorithme exploite généralement plusieurs variables pour traiter des données, celles-ci sont désignées dans l'algorithme par un **nom** unique leur étant attribué par le programmeur. L'unicité des noms de variables permet à l'algorithme d'identifier précisément la variable à être manipulée.

Contrairement à la plupart des langages de programmation traditionnels (tels *C++* et *Java*), le langage *LARP* est *polymorphe contextuel*. Ce terme signifie qu'un algorithme *LARP* n'a pas à définir au préalable ses variables en spécifiant explicitement leur type. Cette philosophie de programmation est couramment exploitée dans les langages de scriptage, tels *Perl* et *Lisp*. Ainsi, le type d'une variable dépend de son contenu, et de ce fait son type peut varier durant l'exécution de l'algorithme.

5.1 Noms de variables

Comme dans tous les langages de programmation, des règles précises régissent la sélection des noms de variables :

- Un nom de variable doit débuter par une lettre (**A** à **Z**, **a** à **z**) ou le caractère de soulignement (**_**).
- Un nom de variable peut être constitué de lettres minuscules (**a** à **z**), de lettres majuscules (**A** à **Z**), de chiffres (**0** à **9**) et du caractère de soulignement (**_**).
- Un nom de variable ne doit pas correspondre à un mot réservé de *LARP*, tels que **ÉCRIRE**, **FIN**, **SI** et **PI**, et ce sans égard aux accents (par exemple **ÉCRIRE** est aussi un mot réservé).
- *LARP* ne fait pas de distinction entre les lettres minuscules et les lettres majuscules, ce qui signifie que **Nom**, **nom** et **NOM** font référence à une même variable.
- *LARP* ignore les accents, ce qui signifie que **Coût**, **Cout** et **Coût** font référence à une même variable.

Voici des exemples de noms de variables valides et invalides :

<u>VALIDES</u>	<u>INVALIDES</u>	
Date	101_Français	Ne commence pas par une lettre ou _
DATE	Cout Achat	L'espace n'est pas accepté
_201	Paie&Bonus	Les caractères spéciaux (autres que _) sont interdits
Table10		
Cout_Achat		
Français		

L'affectation de valeurs aux variables se fait généralement à l'aide de l'[opérateur d'affectation \(=\)](#):

```

\\ Opérateur d'affectation
prix = 10.20           \\ affecte une valeur numérique
nom  = "Gustave Labrie" \\ affecte un chaîne de caractères
notes = [75, 56, 94, 69] \\ affecte un conteneur

```

Pseudo-code 5-1 : Noms de variables

Note : les *conteneurs* sont présentés dans une [section ultérieure](#).

5.2 Opérations

Puisque le langage *LARP* est *polymorphe contextuel*, le type d'une variable dépend de son contenu. Conséquemment le type d'une variable peut varier durant l'exécution de l'algorithme.

En exploitant l'[affectation](#), un algorithme peut emmagasiner une valeur de n'importe quel type dans une variable. De plus, des valeurs de différents types peuvent être combinées dans une même instruction, et la conversion des valeurs d'un type à l'autre (afin d'effectuer l'opération demandée) est généralement transparente.

Par exemple :

```

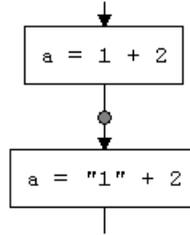
a = 1 + 2           \\ Donne a = l'entier 3
a = "1" + 2        \\ Donne aussi a = l'entier 3
a = "x" + 2        \\ Donne a = la chaîne "x2"
a = "1" + "2"      \\ Donne a = la chaîne "12"

```

Pseudo-code 5-2 : Les opérateurs

Notez dans l'exemple ci-dessus qu'au besoin, *LARP* tente de convertir les chaînes de caractères en valeurs numériques afin d'effectuer l'opération mathématique demandée (ex : **"1"+2** donne **3**). De plus, l'action de certaines opérations sont définies en fonction du type de valeurs auxquelles elles sont appliquées (ex : **"a"+"b"** donne **"ab"**, alors que **1+2** donne **3**).

Les opérations séquentielles telles que celles dans le Pseudo-code 5-2 sont généralement intégrées aux organigrammes à l'aide de l'[instruction séquentielle](#). On retrouve aussi des opérations dans les autres types d'instructions, telles que dans les [conditions](#) et les [arguments d'appel de modules](#).



Organigramme 5-1 : Opérations séquentielles

5.3 Valeurs numériques

Il existe deux formes de valeurs numériques : les *entiers* et *flottants*. Les valeurs entières n'ont pas de partie fractionnelle alors que les valeurs flottantes en ont une.

Voici des exemples de valeurs numériques :

```

Valeur = 2345          \\ Entier en notation décimale
Valeur = 1234.567     \\ Flottant en notation décimale
Valeur = 1.23E-10     \\ Flottant en notation scientifique
  
```

Pseudo-code 5-3 : Les valeurs numériques

La *notation scientifique* permet d'exprimer de très petits nombres (ex : **2.5E-201**) et de très grands nombres (ex : **5E156**). La partie après le **E** (qui peut aussi être écrit en minuscule, **e**) indique la puissance de **10** à multiplier au nombre avant le **E**. Ainsi, **2.1E7** équivaut à **21000000** (i.e. 2.1×10^7), et **2.1E-7** équivaut à **0.00000021** (i.e. 2.1×10^{-7}).

Les limites de valeurs pouvant être manipulées dans un algorithme sont :

- **-2147483648** à **2147483647** pour les valeurs entières, et
- **5.0E-324** (5.0×10^{-324}) à **1.7E308** (1.7×10^{308}) pour les valeurs flottantes.

Toute valeur excédant ces limites dans une instruction (par exemple **1.7E308 * 12**) résulte en une erreur fatale lors de l'exécution de l'algorithme.

5.4 Les chaînes de caractères

Il est possible d'attribuer une chaîne de caractères comme valeur de variable via l'opération d'*affectation*. Il existe deux représentations équivalentes des chaînes de caractères :

```

nom = 'Gustave'       \\ Chaîne spécifiée entre apostrophes
nom = "Gustave"      \\ Chaîne spécifiée entre guillemets
  
```

Pseudo-code 5-4 : La chaîne de caractères

La disponibilité de ces deux représentations permet l'inclusion d'apostrophes ou de guillemets dans les constantes de type chaîne de caractères :

```

phrase = 'Dis "Allo"' \\ Chaîne contenant des guillemets
nom     = "D'Acosta"  \\ Chaîne contenant une apostrophe
  
```

Pseudo-code 5-5 : L'apostrophe et le guillemet

5.5 Séquences d'échappement

LARP traite de façon particulière les *antéslashes* (\) rencontrés dans les chaînes de caractères. L'antéslash indique le début d'une *séquence d'échappement* représentant un caractère spécial. Une séquence d'échappement dans une chaîne de caractères est composée du caractère « \ » suivi d'une lettre spécifique.

Les séquences d'échappement reconnues par *LARP* sont :

Séquence d'échappement	Description
<code>\n</code>	Retour de chariot
<code>\a</code>	Sonnerie (« bip! »)
<code>\b</code>	Retour arrière d'un caractère
<code>\\</code>	Représente le \

Tableau 5-1 : Séquences d'échappement

Notez que la séquence d'échappement `\\` est requise pour représenter l'antéslash dans une chaîne de caractères, puisqu'un seul antéslash est interprété comme étant le début d'une séquence d'échappement.

Voici un exemple d'exploitation de séquences d'échappement. L'instruction ci-dessous active la sonnerie de l'ordinateur puis affiche deux lignes de texte dans la *console d'exécution* :



Organigramme 5-2 : Exemple de séquences d'échappement

5.6 L'affectation

Comme mentionné précédemment, l'*opérateur d'affectation* (=) permet d'assigner une valeur à une variable. Le *nom de la variable* réceptrice est spécifié à gauche de l'opérateur d'affectation, alors qu'à droite de l'opérateur est spécifiée une expression produisant la valeur à être assignée à la variable :

```

a = 123                \\ La variable a reçoit l'entier 123
b = "Bonjour"         \\ La variable b reçoit une chaîne
c = SINUS(10.2) + 1   \\ La variable c reçoit 0.3001253124
  
```

Pseudo-code 5-6 : L'affectation

Puisque *LARP* est polymorphe contextuel, il n'est nul besoin de « déclarer » une variable avant de l'utiliser (comme c'est le cas dans la plupart des langages de programmation, tels *C++* et *Java*). Une variable est automatiquement créée dès qu'on lui attribue une valeur (entre autres via l'opérateur d'affectation).

Lorsque aucune valeur n'est attribuée à une variable, celle-ci est dite *indéfinie*. Si on tente d'afficher la valeur d'une variable indéfinie dans la *console d'exécution*, l'indicateur **#IND** est affiché en rouge pour souligner le fait que la variable n'a pas de valeur attribuée. Un message d'avertissement est aussi affiché dans le *panneau de messages* identifiant quelle variable manipulée dans l'algorithme est indéfinie.

6 Conteneurs

En plus de supporter les [valeurs entières](#), les [valeurs flottantes](#) et les [chaînes de caractères](#), LARP supporte les regroupements de valeurs dans des *conteneurs*. Pour ceux familiers avec les langages de programmation traditionnels (e.g. *C++* et *Java*), un conteneur est une généralisation du *tableau*. Contrairement au tableau traditionnel qui ne peut contenir que des éléments d'un même type (c'est le cas en *C++*), un conteneur LARP peut contenir des données de différents types, incluant des valeurs numériques, des chaînes de caractères et même d'autres conteneurs.

6.1 Regroupement de valeurs

Dans la syntaxe du langage LARP, un *conteneur* est une structure pouvant contenir simultanément plusieurs valeurs. Chaque valeur stockée dans un conteneur peut être [accédée](#), [modifiée](#) et/ou [retirée](#) du conteneur.

Les constantes de type conteneur sont représentées à l'aide des crochets (**[** et **]**) à l'intérieur desquelles les éléments (i.e. les valeurs contenues dans le conteneur) sont énumérées, séparées par la virgule (,). L'exemple qui suit crée des conteneurs et les [affecte](#) à des variables :

```
Jours = ["Lu", "Ma", "Me", "Je", "Ve", "Sa", "Di"]  \\ Conteneur de chaînes
Notes = [45, 78, 56, 96, 35]                 \\ Conteneur de valeurs
                                              \\ entières
```

Pseudo-code 6-1 : Les conteneurs

Les éléments d'un conteneur peuvent être de différents types. Par exemple, le conteneur ci-dessous contient différentes données à propos d'un employé (son nom, son matricule, son salaire, l'année de son embauche et le montant de ses quatre dernières paies). Un conteneur peut même contenir d'autres conteneurs (les quatre paies sont regroupées dans un conteneur) :

```
Dossier = ["Gustave Labrie", 2013345, 56320.00, 1996, $
           [1401.98, 1456.02, 1399.57, 1423.41]]
```

Pseudo-code 6-2 : Conteneur dans un conteneur

Dans l'exemple ci-dessus, le conteneur stocké dans la variable **Dossier** contient 5 éléments, dont le dernier est un conteneur contenant quatre éléments.

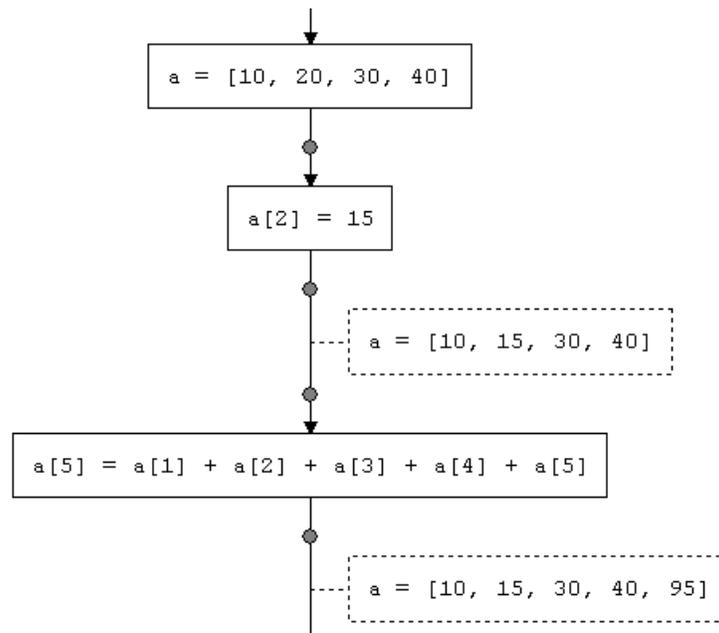
6.2 Accès aux éléments

Les éléments d'un conteneur sont accessibles directement *via leur position* dans le conteneur, cette dernière étant spécifiée entre crochets (**[** et **]**) en suffixe du [nom de la variable](#) conteneur. Le premier élément d'un conteneur est à la position **1**. Dans l'exemple suivant, deux éléments du conteneur affecté à la variable **a** sont accédés (la deuxième instruction accède au premier élément du conteneur, et la dernière instruction accède au troisième élément) :

```
a = [10, 2.3E-12, "Lundi", -17, 0.234]  \\ a est un conteneur
b = a[1] - 3                            \\ b = 7
c = a[3]                                 \\ c = "Lundi"
```

Pseudo-code 6-3 : Accès aux éléments d'un conteneur

Pour modifier les éléments d'un conteneur, on utilise l'**affectation**. On peut aussi utiliser l'affectation pour ajouter des éléments à un conteneur :



Organigramme 6-1 : Modifier les éléments d'un conteneur

6.3 Retirer des éléments

Le fait de faire référence à l'élément d'un conteneur ne retire pas cet élément du conteneur. Ainsi dans l'exemple ci-dessous le conteneur **a** demeure inchangé :

```

ÉCRIRE a[4]
i = 2
b = a[i] + a[i+1]
ÉCRIRE b + a[i-1]
  
```

Pseudo-code 6-4 : Accès aux éléments d'un conteneur

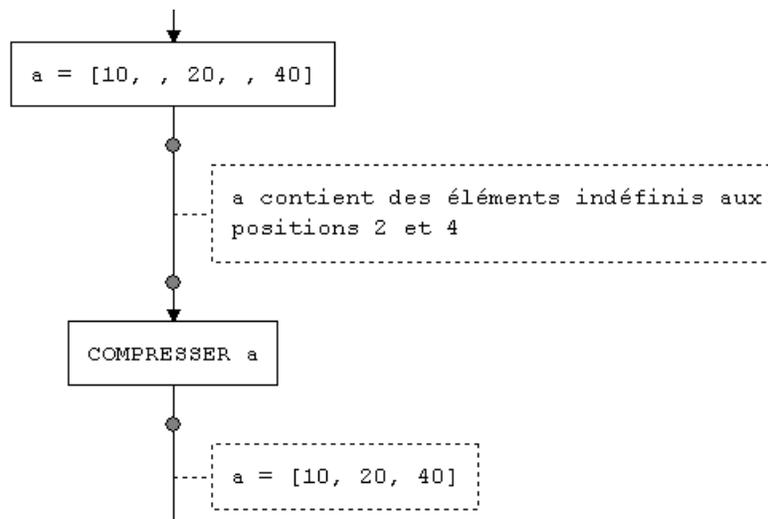
Pour retirer un élément d'un conteneur, il faut faire appel à l'instruction **DÉTRUIRE**. Cette instruction retire l'élément spécifié du conteneur, mais ne libère pas pour autant sa position. La position libérée contient alors un *élément indéfini*. Les valeurs indéfinies sont représentées dans la console d'exécution par l'identificateur **#IND** :

```

a = [10, 20, 30, 40]  \\ a est un conteneur
DÉTRUIRE a[3]       \\ a = [10, 20, , 40]
ÉCRIRE a            \\ affiche [10 20 #IND 40]
a[3] = 30           \\ a = [10, 20, 30, 40]
  
```

Pseudo-code 6-5 : Éléments indéfinis d'un conteneur

L'instruction **COMPRESSER** permet d'éliminer les éléments indéfinis d'un conteneur :



Organigramme 6-2 : Compresser un conteneur

LARP dispose de [fonctions prédéfinies](#) visant à compter les éléments dans un conteneur :

Fonction	Description
CAPACITÉ	Retourne le nombre de positions dans un conteneur, incluant celles n'ayant pas d'élément.
COMPTER	Retourne le nombre d'éléments définis dans un conteneur (i.e. excluant ceux indéfinis).

Tableau 6-1 : Fonctions de manipulation de conteneurs

7 Lecture et écriture

LARP dispose d'instructions pour la **lecture de données (LIRE)** et pour l'**écriture de résultats (ÉCRIRE)**. Ces deux instructions interagissent avec la **console d'exécution** afin de permettre à l'algorithme en exécution de lire des données via le clavier et d'afficher des résultats à l'écran (ces instructions peuvent aussi interagir avec les **fichiers** et les **tampons d'entrées/sorties**). De plus, l'instruction **REQUÊTE** du langage LARP permet simultanément d'afficher une requête et de lire une réponse de l'utilisateur.

LARP offre aussi une **instruction d'entrées/sorties pour organigrammes**. Cette instruction permet de formuler des lectures, des écritures et des requêtes dans un organigramme.

Lorsque plusieurs données sont lues via une même instruction de lecture ou plusieurs résultats sont affichés par une même instruction d'écriture, un caractère de séparation, le **séparateur**, permet de délimiter les valeurs dans la console d'exécution.

Les instructions d'entrées/sorties de LARP sont décrites dans les prochaines sections.

7.1 Instruction d'entrées/sorties pour organigrammes

Les entrées/sorties sont représentées dans les organigrammes LARP par l'**instruction d'entrées/sorties** :

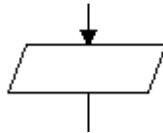


Figure 7-1 : Instruction d'entrées/sorties pour organigrammes

Cette instruction permet de formuler une **lecture**, une **écriture** ou une **requête**. Le type d'instruction représentée dépend des attributs de l'instruction tels que spécifiés dans la **fenêtre d'édition d'instruction d'organigramme** lorsque l'instruction est **éditée** :

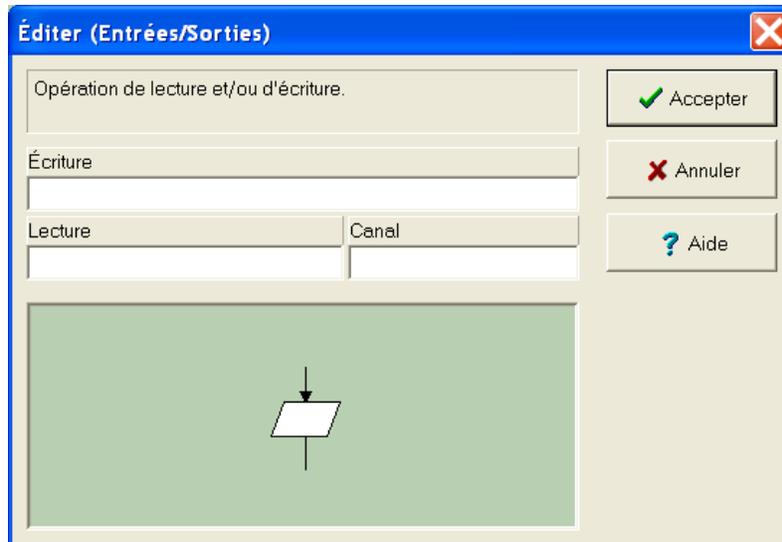


Figure 7-2 : Édition d'une instruction d'entrées/sorties d'organigramme

Si un attribut est spécifié dans la champ **Lecture**, l'instruction d'organigramme en est une de **lecture**. Si un attribut est spécifié dans le champ **Écriture**, c'en est une d'**écriture**. Si les champs **Lecture** et **Écriture** sont tous deux comblés, c'est alors une instruction de **requête**. L'attribut **Canal** permet de spécifier un **canal d'entrées/sorties** afin de rediriger l'instruction vers un **fichier** ou un **tampon d'entrées/sorties**. Notez qu'une instruction de requête peut uniquement être dirigée vers la **console d'exécution** et ne peut conséquemment pas impliquer un canal d'entrées/sorties.

Pour plus d'information sur les attributs d'instructions d'entrées/sorties, consultez les sections suivantes.

7.2 L'instruction de lecture

L'instruction **LIRE** permet de lire une ou plusieurs valeurs (et même des **conteneurs**) tout en les affectant à des variables. Les lectures sont généralement effectuées à partir de la **console d'exécution**, via le clavier de l'ordinateur.

Dans sa plus simple expression, l'instruction **LIRE** lit une unique valeur et l'affecte à la variable spécifiée :

```
LIRE a      \\ Lit une valeur dans la variable a
```

Pseudo-code 7-1 : Lecture d'une valeur

À l'exécution de cette instruction, la console d'exécution affiche un curseur clignotant et attend l'entrée d'une valeur via le clavier. Sitôt le retour de chariot pressé, la valeur entrée est **affectée** à la variable spécifiée (**a**). Le type de valeur affectée à la variable dépend du format du texte entré par l'utilisateur :

Format de l'entrée	Exemples	Type de valeur
Séquence de chiffre	234, -76	<i>Entier</i>
Séquence de chiffres avec décimale ou exposant	2.1, -2E17	<i>Flottant</i>
Séquence de caractères débutant par autre chose qu'un chiffre	Allo, a234	<i>Chaîne</i>

Tableau 7-1 : Interprétation des valeurs lues

L'instruction **LIRE** peut lire plus d'une valeur si une liste de variables lui est fournie :

```
LIRE a, b, c  \\ Lit une valeur pour chaque variable
```

Pseudo-code 7-2 : Lecture de plusieurs valeurs

Lors de la lecture, l'instruction **LIRE** considère l'espace et le retour de chariot comme **séparateurs** de valeurs. Ainsi, à l'exécution de l'exemple ci-dessus et avec les entrées suivantes dans la console d'exécution :

```
Allo 234 -14.78
```

Figure 7-3 : Interprétation du texte fourni lors d'une lecture

Les trois valeurs fournies sont respectivement affectées aux variables correspondantes (**a = "Allo"**, **b = 234** et **c = -14.78**).

Si l'utilisateur ne fournit pas assez de valeurs pour le nombre de variables à lire, l'instruction **LIRE** attend que le nombre requis de valeurs soient fournies avant de poursuivre l'exécution à la

prochaine instruction. Si l'utilisateur fournit trop de valeurs avant le retour de chariot, celles superflues sont ignorées.

Le format de l'attribut **Lecture** d'une instruction d'entrées/sorties dans un organigramme est similaire à celui d'une lecture en pseudo-code :

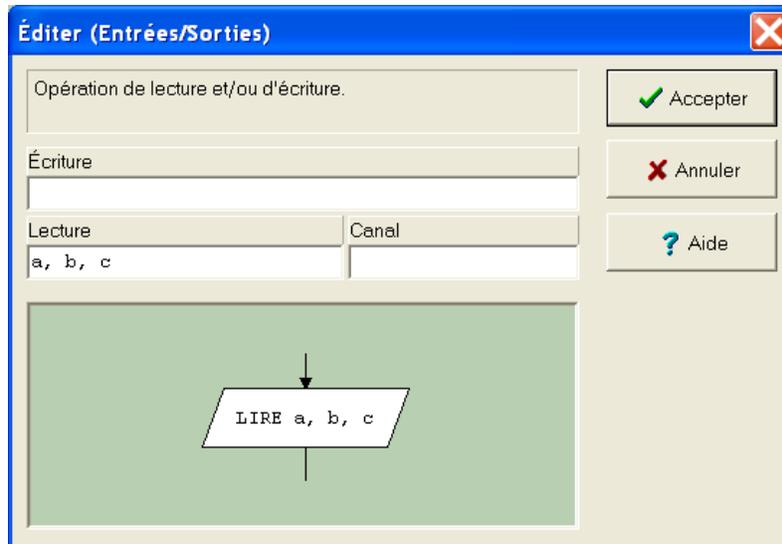


Figure 7-4 : Lecture de valeurs dans un organigramme

7.3 L'instruction d'écriture

L'instruction **ÉCRIRE** permet d'écrire une ou plusieurs valeurs à la [console d'exécution](#). Utilisée seule, l'exécution de cette instruction résulte en l'affichage d'un retour de chariot (i.e. un changement de ligne) dans la console. L'instruction **ÉCRIRE** accepte aussi une ou plusieurs valeurs (ou expressions) à afficher. Lorsque plus d'une valeur sont fournies à l'instruction **ÉCRIRE**, celle-ci les affiche séparées par des espaces (le [séparateur](#) par défaut).

Le Pseudo-code 7-3 utilise l'instruction **ÉCRIRE** pour afficher la valeur d'expressions :

```
a = 100
ÉCRIRE "ALLO"           \\ Écrit une chaîne
ÉCRIRE                \\ Écrit une ligne vide
ÉCRIRE 1, 10+2, a * 2  \\ Écrit trois valeurs
```

Pseudo-code 7-3 : L'instruction d'écriture

Lorsque *LARP* exécute le Pseudo-code 7-3, les résultats suivants sont affichés dans la console d'exécution :

```
ALLO
1 12 200
```

Figure 7-5 : Instruction d'écriture

Notez qu'une instruction **ÉCRIRE** produit toujours un changement de ligne après l'affichage de sa dernière valeur. L'instruction **REQUÊTE** permet d'afficher des résultats sans retour de chariot.

Le format de l'attribut **Écriture** d'une instruction d'entrées/sorties dans un organigramme est similaire à celui d'une écriture en pseudo-code :

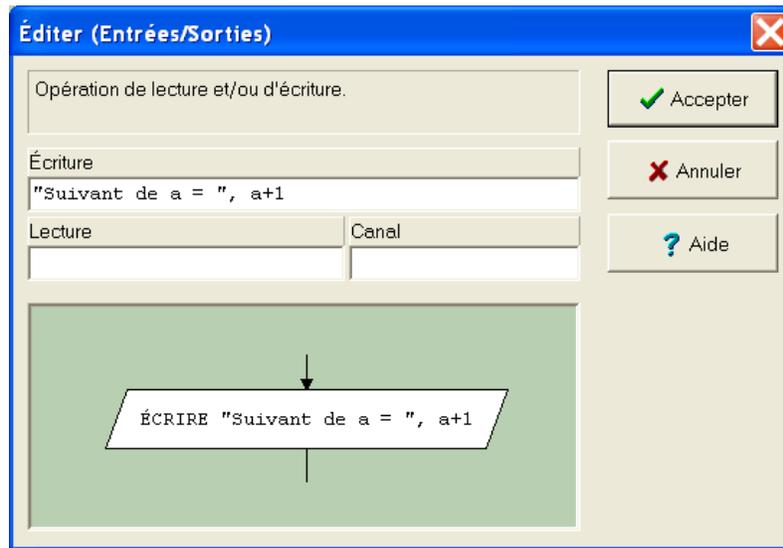


Figure 7-6 : Écriture de valeurs dans un organigramme

7.4 L'instruction de requête

L'instruction **ÉCRIRE** insère automatiquement un changement de ligne après la dernière valeur affichée dans la *console d'exécution*. Ce changement de ligne peut être un inconvénient « esthétique » lorsqu'on veut afficher une requête et lire une ou plusieurs valeurs sur une même ligne.

Considérons le pseudo-code suivant :

```
ÉCRIRE "Entrez un nombre: "  \\ Libellé de la requête
LIRE Nombre
```

Pseudo-code 7-4 : Interroger l'utilisateur

Ce pseudo-code affiche le libellé de la requête sur une ligne, puis effectue la lecture du nombre sur la ligne suivante (car l'instruction **ÉCRIRE** termine toujours l'affichage des valeurs avec un retour de chariot) :

```
Entrez un nombre:
17
```

Figure 7-7 : Interrogation peu élégante

L'instruction **REQUÊTE** évite ce désagrément en permettant d'afficher un libellé et d'effectuer une lecture dans une même instruction :

```
REQUÊTE "Entrez un nombre: ", Nombre  \\ Libellé et lecture
```

Pseudo-code 7-5 : Instruction REQUÊTE

L'instruction ci-dessus affiche le libellé de la requête et effectue la lecture du nombre sur une même ligne :

```
Entrez un nombre: 17
```

Figure 7-8 : Instruction REQUÊTE

Notez que, tout comme l'instruction **ÉCRIRE**, l'instruction **REQUÊTE** peut obtenir la chaîne composant le libellé à partir de l'évaluation d'une expression. L'instruction **REQUÊTE** peut aussi lire plus d'une valeur, tout comme l'instruction **LIRE** :

```
Libellé = "Entrez trois nombres: "
REQUÊTE Libellé, N1, N2, N3          \\ Libellé via une variable
```

Pseudo-code 7-6 : Une requête pour lire plus d'une valeur

La lecture de valeurs est optionnelle dans une instruction **REQUÊTE**. Si seul un libellé est fourni, l'instruction **REQUÊTE** affiche cette chaîne de caractères sans changement de ligne et sans effectuer une lecture.

Le format des attributs **Écriture** et **Lecture** d'une instruction d'entrées/sorties dans un organigramme est similaire à celui d'une instruction **REQUÊTE** en pseudo-code :

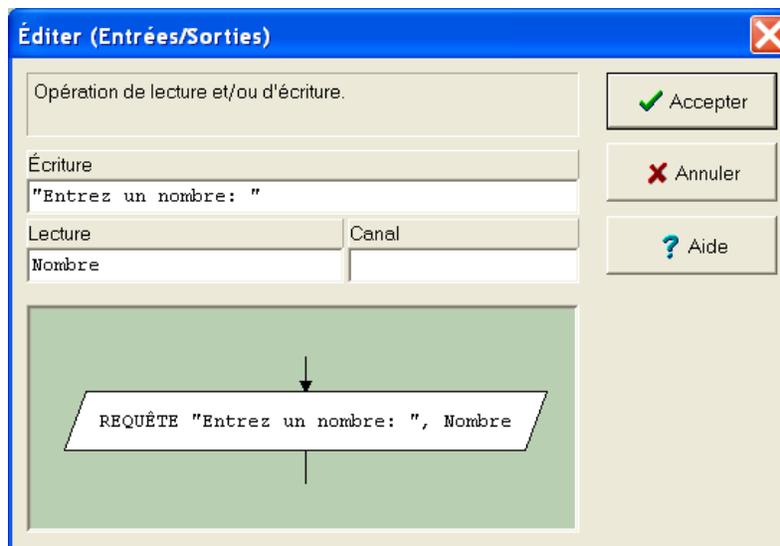


Figure 7-9 : Formuler une requête dans un organigramme

7.5 Le séparateur

Par défaut, les instructions **LIRE** et **REQUÊTE** utilisent le caractère d'espace pour séparer les valeurs lors de la lecture via la console d'exécution, et l'instruction **ÉCRIRE** (ainsi que **REQUÊTE**) utilise aussi le caractère d'espace pour séparer les valeurs écrites dans la console d'exécution :

Instruction	Interprétation par défaut des espaces
LIRE	Les espaces permettent à l'instruction de distinguer une valeur de la suivante.
ÉCRIRE	Un espace sépare les valeurs affichées par une même instruction. ÉCRIRE utilise aussi l'espace pour séparer les éléments d'un conteneur à l'affichage.

Tableau 7-2 : Interprétation des espaces par LIRE et ÉCRIRE

LARP dispose d'une instruction permettant de modifier le caractère utilisé comme séparateur :

ÉCRIRE 10, 20	\\ Écrit deux valeurs séparées par un espace
SÉPARATEUR ", "	\\ Changement de séparateur
ÉCRIRE 10, 20	\\ Écrit deux valeurs séparées par une virgule

Pseudo-code 7-7 : Instruction **SÉPARATEUR**

Les instructions ci-dessus affichent dans la console d'exécution :

```
10 20
10,20
```

Figure 7-10 : Changement de séparateur

Le séparateur a aussi un impact sur l'instruction **LIRE** : l'utilisateur doit entrer le séparateur actif afin de distinguer les données entrées. Par exemple, considérons le pseudo-code suivant :

ÉCRIRE "Nom?"	
LIRE Nom	
ÉCRIRE "Nom lu = ", Nom	
SÉPARATEUR ", "	\\ Changement de séparateur
ÉCRIRE "Nom?"	
LIRE Nom	
ÉCRIRE "Nom relu = ", Nom	

Pseudo-code 7-8 : Modification du séparateur

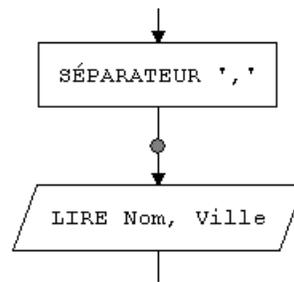
Si l'utilisateur fournit la chaîne **Gustave Labrie** pour chaque instruction de lecture lors de l'exécution de ce pseudo-code, celui-ci affiche dans la console d'exécution :

```
Nom?
Gustave Labrie
Nom lu = Gustave
Nom?
Gustave Labrie
Nom relu = Gustave Labrie
```

Figure 7-11 : Lecture exploitant le séparateur

L'utilisation d'un séparateur alternatif permet ainsi à un algorithme de lire une chaîne contenant des espaces. Notez que le retour de chariot est toujours considéré comme un séparateur, quel que soit le caractère spécifié comme séparateur actif avec l'instruction **SÉPARATEUR**.

Dans un organigramme, l'instruction **SÉPARATEUR** doit être insérée à l'intérieur d'une instruction séquentielle :



Organigramme 7-1 : Changement de séparateur

8 Opérateurs et fonctions prédéfinies

Afin de permettre les calculs mathématiques, *LARP* supporte les opérations arithmétiques de base et les principales fonctions mathématiques généralement retrouvées dans les langages de programmation traditionnels tels *Java* et *C++*. *LARP* offre aussi plusieurs fonctions servant à la manipulation de [chaînes de caractères](#) et de [conteneurs](#).

8.1 Opérateurs arithmétiques

LARP supporte les principaux opérateurs arithmétiques. Ceux-ci sont présentés ci-dessous, en ordre croissant de priorités (les opérateurs de priorité commune sont énumérés sur une même ligne) :

Opérateurs	Description
+, -	Addition et soustraction
*, /, //, %	Multiplication, division flottante, division entière et modulo
^	Puissance
-	Négation

Tableau 8-1 : Opérateurs arithmétiques

La *division entière* est la division d'une valeur entière par une autre valeur entière et donnant un résultat entier; s'il y a un reste, celui-ci est ignoré. Ainsi, **17//5** donne **3**, le reste ignoré étant **2**. L'*opérateur modulo* (%) permet d'obtenir le reste d'une division entière. Par exemple, **17%5** donne 2, soit le reste de **17//5**.

Dans une expression, *LARP* évalue donc d'abord - (la négation), puis ^, puis *, /, // et %, et enfin + et -. Pour deux opérateurs successifs de priorité égale, l'évaluation se fait de la gauche vers la droite. Les expressions peuvent être regroupées à l'intérieur de parenthèses afin de modifier l'ordre d'évaluation de celles-ci :

ÉCRIRE	7/5	\\ affiche 1.4
ÉCRIRE	7//5	\\ affiche 1
ÉCRIRE	7%5	\\ affiche 2
ÉCRIRE	2^5	\\ affiche 32
ÉCRIRE	4+8/2+1	\\ équivaut à 4+(8/2)+1 = 9
ÉCRIRE	(4+8)/(2+1)	\\ affiche 4

Pseudo-code 8-1 : Les opérateurs arithmétiques

L'[animation](#) peut être exploitée en [exécution pas-à-pas](#) pour visualiser l'impact des priorités d'évaluation des opérateurs arithmétiques dans une expression.

Certains opérateurs arithmétiques n'acceptent qu'un type de valeurs. Ainsi, la division entière (//) et le module (%) convertissent au besoin les deux expressions chaque côté de l'opérateur en valeurs entières. Lorsqu'une telle conversion à un entier est requise, la partie fractionnelle de la valeur flottante est éliminée. Ainsi, **14.8%5** deviendra lors de l'évaluation **14%5**, donnant **4**.

Lorsqu'une expression arithmétique implique des types non compatibles, *LARP* interrompt l'exécution de l'algorithme et affiche un [message d'erreur](#) indiquant l'incompatibilité. Une telle incompatibilité est aussi rapportée lorsqu'une [valeur indéfinie](#) est impliquée dans l'évaluation d'une expression.

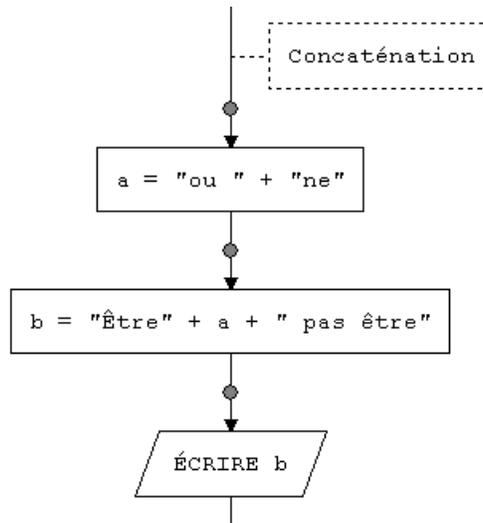
8.2 Opérateurs de chaînes de caractères

LARP supporte un seul opérateur applicable aux chaînes de caractères :

Opérateur	Description
+	Concaténation (i.e. joindre deux chaînes)

Tableau 8-2 : Opérateurs de chaînes de caractères

L'opérateur `+` est utilisé pour joindre des chaînes bout à bout (communément appelée la *concaténation*). L'exemple ci-dessous démontre l'utilisation de cet opérateur (l'organigramme affiche la chaîne *Être ou ne pas être*) :



Organigramme 8-1 : La concaténation de chaînes de caractères

Notez que les chaînes de caractères impliquées dans l'opération demeurent inchangées. Ainsi dans la deuxième instruction d'affectation de l'Organigramme 8-1, la chaîne dans la variable `a` est inchangée.

Attention : lorsque l'opérateur `+` implique une chaîne de caractères et un nombre (entier ou flottant), *LARP* tente d'abord de convertir la chaîne de caractères en nombre afin d'effectuer une addition. Si la tentative de conversion échoue, le nombre est converti en chaîne de caractères puis il y a concaténation. Les exemples qui suivent démontrent ces conversions de type :

<code>a = "12" + 10</code>	<code>\\ a = l'entier 22</code>
<code>b = 10 + "12"</code>	<code>\\ b = l'entier 22</code>
<code>c = "12z" + 10</code>	<code>\\ c = la chaîne "12z10"</code>
<code>d = "12" + "10"</code>	<code>\\ d = la chaîne "1210"</code>
<code>e = d + 2000</code>	<code>\\ e = l'entier 3210</code>
<code>f = "12" + "10" + 5</code>	<code>\\ f = l'entier 1215, car le premier +</code>
	<code>\\ produit "1210"</code>

Pseudo-code 8-2 : L'opérateur `+` appliqué à des valeurs de type distincts

8.3 Opérateurs de conteneurs

LARP supporte deux opérateurs applicables aux [conteneurs](#) :

Opérateur	Description
+	Concaténation (i.e. joindre deux conteneurs).
-	Différence, éliminant d'un conteneur les éléments retrouvés dans un second conteneur.

Tableau 8-3 : Opérateurs de conteneur

L'opérateur **+**, communément appelé la *concaténation*, est utilisé pour joindre deux conteneurs bout à bout ou ajouter un élément au début ou à la fin d'un conteneur. L'exemple qui suit démontre l'utilisation de cet opérateur :

```
a = [20, 30] + [30, 40]      \ \ a = [20, 30, 30, 40]
b = 10 + a + 50            \ \ b = [10, 20, 30, 30, 40, 50]
```

Pseudo-code 8-3 : Joindre des conteneurs

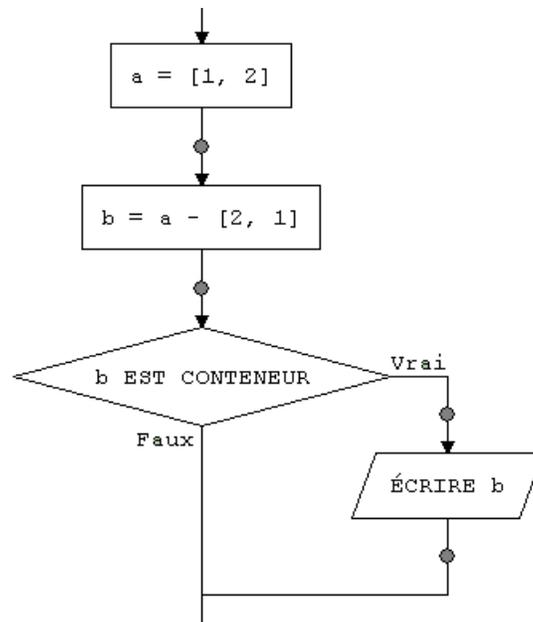
L'opérateur **-** élimine du premier conteneur tout élément retrouvé dans le second conteneur. On peut aussi spécifier une valeur plutôt qu'un second conteneur :

```
a = [1, 2, 3, 2, 4] - [2, 5]  \ \ a = [1, 3, 4]
b = a - 3                    \ \ b = [1, 4]
```

Pseudo-code 8-4 : Soustraction de conteneurs

Notez que les conteneurs impliqués dans l'opération demeurent inchangés. Ainsi dans la deuxième ligne du Pseudo-code 8-4, le conteneur dans la variable **a** demeure inchangé.

Attention : lorsque l'opérateur **-** élimine tous les éléments d'un conteneur, la valeur retournée est [indéfinie \(#IND\)](#). On peut utiliser le mot réservé **CONTENEUR** dans un [test de type](#) pour déterminer si l'opération résulte en un conteneur. Dans l'exemple suivant rien ne sera affiché puisque la différence produit un résultat indéfini (i.e. aucun conteneur résultant) :



Organigramme 8-2 : Test de type CONTENEUR

8.4 Fonctions prédéfinies

Les fonctions prédéfinies de *LARP* sont catégorisées selon les types de valeurs auxquelles elles sont applicables :

1. les [fonctions mathématiques](#), applicables aux valeurs numériques,
2. les [fonctions de manipulation de chaînes de caractères](#), et
3. les [fonctions de manipulation de conteneurs](#).

Les sections suivantes décrivent brièvement les fonctions de chaque catégorie. Pour obtenir plus d'information ou connaître la syntaxe d'invocation de ces fonctions, consultez l'[Annexe C](#).

8.4.1 Fonctions mathématiques

Voici les principales constantes et fonctions mathématiques offertes dans le langage *LARP* :

Fonctions	Descriptions	Exemples
ABSOLU	Retourne la valeur absolue de la valeur donnée.	ABSOLU (-6)
ALÉATOIRE	Retourne un nombre flottant ou entier choisi au hasard (plusieurs versions de la fonction sont disponibles).	ALÉATOIRE ALÉATOIRE (11) ALÉATOIRE (2.3, 15.0)
ARCTANGENTE	Retourne \tan^{-1} de la valeur donnée en radians.	ARCTANGENTE (0.0)
ARRONDIR	Retourne la valeur donnée arrondie au plus proche entier.	ARRONDIR (12.6) <i>retourne 13</i>

Fonctions	Descriptions	Exemples
COSINUS	Retourne le cosinus de la valeur donnée en radians.	COSINUS(1.5707963)
ENCHAÎNE	Convertit la valeur donnée en chaîne de caractères.	ENCHAÎNE(12.34)
EXP	Retourne la base du logarithme naturel (e).	LOGE(EXP) <i>retourne 1</i>
LOG10	Retourne le logarithme en base 10 de la valeur donnée.	LOG10(100) <i>retourne 2</i>
LOGE	Retourne le logarithme base e de la valeur donnée.	LOGE(2.1)
MAXIMUM	Retourne la plus grande valeur parmi celles données (deux valeurs ou plus)	MAXIMUM(11.1, 12, 7)
MINIMUM	Retourne la plus petite valeur parmi celles données (deux valeurs ou plus).	MINIMUM(11.1, 12, 7)
PI	Retourne la valeur de la constante mathématique Pi.	aire = PI * r ^ 2
PLAFOND	Retourne le plus petit entier supérieur ou égal à la valeur donnée.	PLAFOND(12.1) <i>retourne 13</i>
PLANCHER	Retourne le plus grand entier inférieur ou égal à la valeur donnée.	PLANCHER(12.1) <i>retourne 12</i>
RACINE	Retourne la racine carrée de la valeur donnée.	RACINE(25) <i>retourne 5</i>
SINUS	Retourne le sinus de la valeur donnée en radians.	SINUS(1.5707963)

Tableau 8-4 : Fonctions mathématiques prédéfinies

8.4.2 Fonctions de manipulation de chaînes

LARP dispose de plusieurs fonctions prédéfinies destinées à la manipulation de chaînes de caractères. Malgré le nombre limité de fonctions, celles-ci offrent l'essentiel de la fonctionnalité requise afin d'écrire des modules de manipulation de chaînes sophistiqués :

Fonctions	Descriptions	Exemple
COMPTER	Retourne le nombre de caractères dans la chaîne (synonyme de LONGUEUR).	COMPTER("allo") <i>retourne 4</i>
ENCARACTÈRES	Convertit la chaîne de caractères donnée en un conteneur ayant chaque caractère comme élément distinct.	ENCARACTÈRES("Bye") <i>retourne ['B', 'y', 'e']</i>

Fonctions	Descriptions	Exemple
FORMATER	Retourne une chaîne de caractères formée à partir d'une chaîne de formatage et d'une séquence d'arguments.	FORMATER ("%5.2f", 3.1) <i>retourne " 3.10"</i>
LONGUEUR	Retourne le nombre de caractères dans la chaîne (synonyme de COMPTER).	LONGUEUR ("allo") <i>retourne 4</i>
MAJUSCULES	Retourne la chaîne donnée avec ses lettres minuscules converties en majuscules.	MAJUSCULES ("Allo") <i>retourne "ALLO"</i>
MINUSCULES	Retourne la chaîne donnée avec ses lettres majuscules converties en minuscules.	MINUSCULES ("Allo") <i>retourne "allo"</i>
POSITION	Retourne la position où débute la première occurrence de la première chaîne dans la seconde.	POSITION ("cd", "abcde") <i>retourne 3</i>
SOUSENSEMBLE	Retourne un sous-ensemble de la chaîne fournie (le 2 ^{ème} paramètre indique l'index de départ, et le 3 ^{ème} indique le nombre de caractères à extraire).	SOUSENSEMBLE ("abcde", 2, 3) <i>retourne "bcd"</i>

Tableau 8-5 : Fonctions prédéfinies de manipulation de chaînes de caractères

8.4.3 Fonctions de manipulation de conteneurs

LARP dispose de plusieurs fonctions prédéfinies destinées à la manipulation de **conteneurs** :

Fonctions	Descriptions	Exemple
CAPACITÉ	Retourne le nombre d'éléments définis et non définis (voir la commande DÉTRUIRE) dans le conteneur.	CAPACITÉ ([1, , 2,]) <i>retourne 4</i>
COMPTER	Retourne le nombre d'éléments définis dans le conteneur.	COMPTER ([1, , 2,]) <i>retourne 2</i>
ENCARACTÈRES	Convertit récursivement les éléments du conteneur donné en un conteneur ayant chaque caractère comme élément distinct.	ENCARACTÈRES ("ab", 2) <i>retourne ['a', 'b', '2']</i>
ENCHAÎNE	Convertit les éléments du conteneur en chaînes de caractères et joint celles-ci ensembles.	ENCHAÎNE ([10, "x"]) <i>retourne "10x"</i>
MAXIMUM	Retourne la plus grande valeur parmi celles du conteneur.	MAXIMUM ([11, 12, 7])
MINIMUM	Retourne la plus petite valeur parmi celles du conteneur.	MINIMUM ([11, 12, 7])

Fonctions	Descriptions	Exemple
POSITION	Retourne la position de la première occurrence de la valeur donnée dans le conteneur.	POSITION(8, [1, 8, 5]) <i>retourne 2</i>
SOUSENSEMBLE	Retourne un sous-ensemble du conteneur donné (le 2 ^{ième} paramètre indique l'index de départ, et le 3 ^{ième} indique le nombre d'éléments à extraire).	SOUSENSEMBLE([1, 4, 9, 5, 11], 2, 3) <i>retourne [4, 9, 5]</i>

Tableau 8-6 : Fonctions prédéfinies de manipulation de conteneurs

9 Structures conditionnelles

Les algorithmes présentés dans les sections précédentes sont constitués d'instructions exécutées séquentiellement, de l'instruction **Début** jusqu'à l'instruction **Fin**.

La plupart des problèmes à résoudre par programmation comporte l'éventualité d'effectuer un choix dans l'algorithme. Une *structure conditionnelle* est une instruction permettant de spécifier des séquence d'instructions alternatives dans un algorithme.

LARP offre quatre structures conditionnelles :

1. La *structure SI*
2. La *structure SI-SINON*
3. La *structure SI-SINON-SI*
4. La structure de *SÉLECTION*

Dans sa forme la plus simple (la structure SI), une structure conditionnelle sous forme pseudo-code est composée des mots réservés **SI**, **ALORS** et **FINSI**, d'une *condition* et d'une *séquence d'instructions* à exécuter lorsque la condition est vraie. Dans un organigramme la structure SI est composée d'un instruction conditionnelle où la *séquence d'instructions* est associée à la branche étiquetée *Vrai* :

Structure conditionnelle SI	
<p>SI <i>condition</i> ALORS <i>Séquence d'instructions</i> FINSI</p>	<pre> graph TD Start(()) --> Condition{condition} Condition -- Vrai --> Seq[Séquence d'instructions] Condition -- Faux --> End(()) </pre>

Tableau 9-1 : La structure conditionnelle SI

Dans la structure pseudo-code ci-dessus (à gauche dans le Tableau 9-1), le mot réservé **SI** indique le début de la structure conditionnelle, et le mot réservé **FINSI** en indique la fin. Dans la structure en organigramme correspondante (à droite dans le Tableau 9-1) la condition indique le début de la structure conditionnelle et la convergence des deux branches (branches *Vrai* et *Faux*) en indique la fin. Notez que LARP permet d'orienter la branche *Vrai* à droite ou à gauche de la condition.

Les structures conditionnelles sont basées sur l'évaluation d'une *condition*, dont le résultat est *vrai* ou *faux*. C'est sur la base de cette condition que le flux d'exécution est déterminé.

9.1 Les conditions

Une condition est une comparaison. Cet énoncé décrit l'essentiel de ce qu'est une condition. En pratique, une *condition simple* est composée d'au moins trois éléments :

1. une première valeur,
2. un opérateur de comparaison, et
3. une seconde valeur.

Les valeurs peuvent être a priori de n'importe quel type ([numériques](#), [chaînes de caractères](#) ou [conteneurs](#)) et peuvent être spécifiées explicitement sous forme de [constantes](#) ou implicitement sous forme d'expressions à évaluer. Si l'on veut que la comparaison ait un sens, il faut cependant que les deux valeurs comparées soient du même type ou de types comparables.

L'opérateur de comparaison dans une condition simple est appelé un [opérateur relationnel](#). Ces opérateurs permettent de comparer l'envergure de deux valeurs.

Une condition simple peut aussi être un [test de type](#) : ce test sert à vérifier le type de la valeur résultant de l'évaluation d'une expression.

Enfin, des conditions simples peuvent être regroupées en une *condition composée* à l'aide des [opérateurs logiques](#).

9.2 Opérateurs relationnels

Les opérateurs relationnels de *LARP* permettent de comparer deux valeurs :

Opérateurs	Descriptions
<	Plus petit : $a < b$
<=	Plus petit ou égal : $a <= b$
>	Plus grand : $a > b$
>=	Plus grand ou égal : $a >= b$
=	Égalité : $a = b$
!=	Inégalité : $a != b$. Le symbole équivalent <> est aussi supporté par <i>LARP</i> .

Tableau 9-2 : Opérateurs relationnels

Les opérateurs relationnels permettent de comparer deux valeurs de types comparables. Des valeurs sont de types comparables lorsqu'elles peuvent être logiquement comparées. Ainsi, alors qu'il est logique de comparer une valeur entière à une valeur flottante, ça ne l'est pas de comparer une valeur entière à un [conteneur](#).

Voici des exemples de [conditions simples](#) :

```
\\ Lire deux valeurs
ÉCRIRE "Entrez deux valeurs: "
LIRE a, b

\\ Identifier la plus petite de deux valeurs lues
SI a < b ALORS
  ÉCRIRE "Minimum = ", a
FINSI
SI a >= b ALORS
  ÉCRIRE "Minimum = ", b
FINSI

\\ Déterminer si une des deux valeurs est 0
SI a*b = 0 ALORS
  ÉCRIRE "Au moins une valeur est 0"
  ÉCRIRE "Veuillez entrer de nouvelles valeurs: "
  LIRE a, b
FINSI
```

Pseudo-code 9-1 : Les conditions simples

Comme le démontre l'exemple ci-dessus, la condition simple peut en fait être composée d'expressions, et la séquence d'instructions dans la structure conditionnelle peut être constituée d'une ou plusieurs instructions.

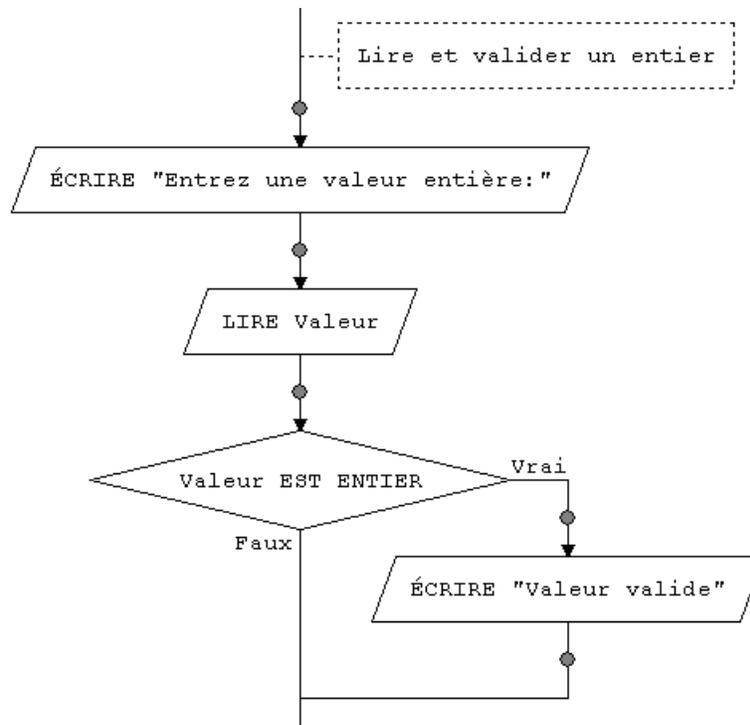
Notez que les opérateurs relationnels peuvent très bien être employés pour comparer des [chaînes de caractères](#) ou des conteneurs :

- Lorsque deux chaînes de caractères sont comparées, celles-ci le sont selon l'ordre alphabétique en fonction du [codage ASCII](#). Ainsi, "abc" < "b", mais "abc" > "B".
- L'égalité de conteneurs est déterminée selon leurs éléments. La comparaison est [récursive](#) lorsque des éléments sont eux-mêmes des conteneurs. Seuls les opérateurs = sont != sont applicables aux conteneurs.

Attention : les opérateurs relationnels ne peuvent pas être enchaînés. Par exemple, la condition $5 < a < 10$ est invalide. Il faut exploiter les *opérateurs logiques* pour exprimer de telles conditions.

9.3 Tests de type

Il est parfois nécessaire de vérifier le type d'une valeur avant de procéder à son traitement. C'est ainsi le cas lorsqu'on veut valider une valeur entrée par l'utilisateur. LARP dispose des mots réservés **ENTIER**, **FLOTTANT**, **CHAÎNE** et **CONTENEUR** qui, utilisés conjointement avec le mot réservé **EST**, permettent de vérifier le type de la valeur résultante de l'évaluation d'une expression :



Organigramme 9-1 : Condition testant le type du contenu d'une variable

Les mots réservés **DÉFINIE** et **INDÉFINIE**, utilisés aussi avec le mot réservé **EST**, permettent de vérifier si une variable ou un élément de conteneur est défini :

```

SI Tab[1] EST INDÉFINIE ALORS
  COMPRESSER Tab
FINSI
  
```

Pseudo-code 9-2 : Condition vérifiant si une valeur est définie ou non

Notez que *LARP* reconnaît les mots réservés **DÉTERMINÉE** et **INDETERMINÉE** comme étant respectivement des synonymes de **DÉFINIE** et **INDÉFINIE**. Notez aussi que ces quatre mots réservés peuvent aussi bien être écrits au masculin (i.e. **DÉFINI**, **INDETERMINÉ**) qu'au féminin.

9.4 Opérateurs logiques

Les *opérateurs logiques* permettent de relier des *conditions simples* en une seule « super-condition ». Le regroupement de conditions est parfois requis pour spécifier qu'un ensemble de conditions doit être satisfait pour procéder à l'exécution d'une séquence d'instructions. Par exemple, une *condition composée* est requise pour exprimer la condition « *la valeur doit être supérieure à zéro et inférieure à 100* » ou « *la couleur doit être rouge ou verte* ». Les opérateurs logiques permettent de tels regroupements de conditions simples.

Trois opérateurs logiques permettent de regrouper des conditions. Ceux-ci sont :

Opérateurs	Descriptions
ET	Les deux conditions doivent être satisfaites : $a > 0$ ET $a < 100$
OU	Au moins une des deux conditions doit être satisfaite : $a < 1$ OU $a > 99$
!	Négation logique de la condition : $!(a \text{ EST ENTIER})$. Le mot réservé NON est équivalent.

Tableau 9-3 : Opérateurs logiques

Notez que l'opérateur logique de négation est employé avec les parenthèses; celles-ci permettent de préciser la condition devant être inversée.

Contrairement aux [opérateurs relationnels](#), les opérateurs logiques peuvent être enchaînés :

```

\\ Lire et valider une couleur
ÉCRIRE "Entrez une couleur: "
LIRE couleur
SI couleur="bleu" OU couleur="blanc" OU couleur="rouge" ALORS
  ÉCRIRE "Couleur invalide"
FINSI

```

Pseudo-code 9-3 : Les conditions composées

9.5 Priorité des opérateurs

Comme discuté dans la section portant sur les [opérateurs arithmétiques](#), tous les opérateurs de *LARP* ont un niveau de priorité leur étant attribué. La priorité d'un opérateur a un impact sur l'ordre d'évaluation des composants d'une expression ou d'une [condition](#). Le Tableau 9-4 présente les opérateurs de *LARP* en ordre croissant de priorité; les opérateurs de même priorité sont regroupés sur une même ligne :

Opérateurs	Descriptions
OU	Ou logique
ET	Et logique
NON	Négation logique (le symbole ! est équivalent).
<, <=, >, >=, =, !=	Opérateurs relationnels
+, -	Addition et soustraction
*, /, //, %	Multiplication, division flottante, division entière et modulo
^	Puissance
-	Négation arithmétique

Tableau 9-4 : Priorité des opérateurs

Dans une expression ou une condition, les priorités d'opérateurs peuvent être circonscrites avec l'emploi des parenthèses.

Voici un exemple d'ordre d'évaluation d'une condition composée en fonction de la priorité des opérateurs impliqués :

$$! a+2 < 30 \text{ OU } b-c\%2 = 28 \text{ ET } c^{11} > 2000*c+1$$

Figure 9-1 : Condition composée

Cette condition est équivalente à :

$$(! ((a+2) < 30)) \text{ OU } (((b-(c\%2)) = 28) \text{ ET } ((c^{11}) > ((2000*c)+1)))$$

Figure 9-2 : Ordre d'évaluation de la condition précédente

L'évaluation de la condition est donc effectuée ainsi :

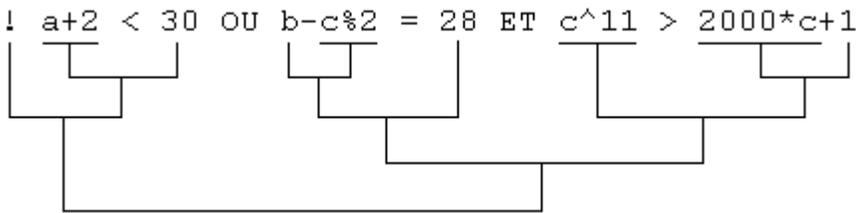


Figure 9-3 : Représentation graphique de l'ordre d'évaluation

L'animation peut être exploitée en [exécution pas-à-pas](#) pour visualiser l'impact des priorités d'évaluation des opérateurs dans une expression ou une condition.

Structures SI et SI-SINON

Il n'y a que deux formes possibles de structures SI : la forme de droite dans le Tableau 9-5 est la forme complète, celle de gauche la forme simple.

Structure SI			Structure SI-SINON		
SI	<i>condition</i>	ALORS	SI	<i>condition</i>	ALORS
<i>Séquence</i>	<i>d'instructions</i>	#1	<i>Séquence</i>	<i>d'instructions</i>	#1
FINSI			SINON		
			<i>Séquence d'instructions #2</i>		
			FINSI		

Tableau 9-5 : Structures conditionnelles SI et SI-SINON

Une *condition* est une expression composée d'*opérateurs relationnels* (parfois aussi d'*opérateurs arithmétiques* et *logiques*) dont la valeur est vraie ou fausse. Cela peut donc être :

- une condition, ou
- un test de type.

Ces deux structures conditionnelles sont relativement claires. Lorsque le flux d'exécution atteint la structure (i.e. la ligne **SI condition ALORS** du pseudo-code, ou le losange de la condition dans l'organigramme), LARP examine la valeur de la *condition*. Si cette condition est vraie, la *Séquence d'instructions #1* est exécutée. Cette séquence d'instructions peut être constituée d'une seule instruction ou de plusieurs, cela n'a aucune importance. À la fin de la *Séquence d'instructions #1*, l'exécution se poursuit à la fin de la structure conditionnelle :

- Au moment où le flux d'exécution arrive au mot **SINON** du pseudo-code, LARP saute directement à la première instruction située après le **FINSI**.
- Au moment où le flux d'exécution atteint le point de convergence des branchements de la structure conditionnelle de l'organigramme, LARP quitte cette structure.

Réciproquement, si la *condition* est fausse, le flux d'exécution ignore la *Séquence d'instructions #1* et passe directement à la première ligne située après le **SINON** afin d'exécuter la *Séquence d'instructions #2* dans le cas du pseudo-code, ou après le point de convergence des branchements de la structure conditionnelle de l'organigramme.

```

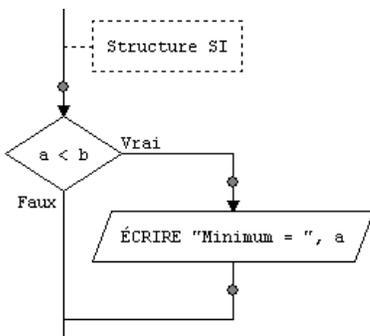
\\ Structure SI
SI a < b ALORS
  ÉCRIRE "Minimum = ", a
FINSI

\\ Structure SI-SINON
SI a < b ALORS
  ÉCRIRE "Minimum = ", a
SINON
  ÉCRIRE "Minimum = ", b
FINSI

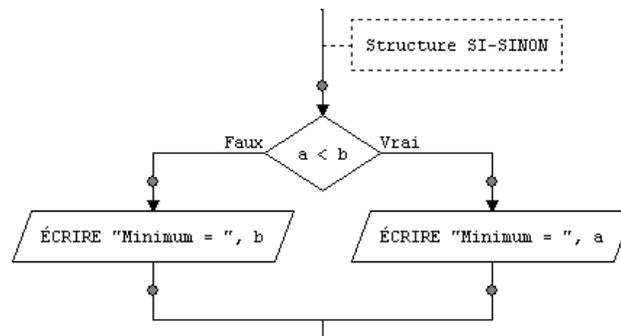
```

Pseudo-code 9-4 : Structures SI et SI-SINON

La forme simplifiée correspond au cas où la branche fautive du **SI** est vide. Dès lors, plutôt qu'écrire « sinon ne rien faire du tout », il est plus simple de ne rien écrire.



Organigramme 9-2 : Structure SI



Organigramme 9-3 : Structure SI-SINON

Notez que *LARP* permet d'orienter la branche *Vrai* à droite ou à gauche de la condition dans un organigramme. Dans le cas de la structure SI-SINON, la branche *Faux* est évidemment orientée à l'opposé de la branche *Vrai*.

Puisque *LARP* utilise le changement de ligne pour distinguer les instructions dans le pseudo-code, il est important de respecter la syntaxe de *LARP* dans les structures conditionnelles. Ainsi, le pseudo-code suivant n'est pas accepté par *LARP* :

```

\\ Syntaxe erronée (il manque un changement de ligne après ALORS)
SI a < b ALORS ÉCRIRE "Minimum = ", a
FINSI

\\ Syntaxe erronée (il y a un changement de ligne superflu avant ALORS)
SI a < b
ALORS
    ÉCRIRE "Minimum = ", a
FINSI

```

Pseudo-code 9-5 : Syntaxes invalides de la structure SI

Notez que l'éditeur graphique de *LARP* ne permet pas de formuler une structure conditionnelle invalide dans un organigramme.

9.7 Structures SI-SINON imbriquées

Graphiquement, on peut très facilement imaginer une structure conditionnelle SI-SINON comme un aiguillage de chemin de fer : cette structure conditionnelle ouvre deux voies, correspondant à deux flux d'exécution différents. Mais il y a des situations où deux voies ne suffisent pas. Par exemple, un programme devant donner l'état de l'eau selon sa température peut devoir choisir entre trois réponses possibles (solide, liquide ou gazeuse).

Une première solution est la suivante :

```

ÉCRIRE "Température de l'eau? "
LIRE Temp
SI Temp <= 0 ALORS                                \\ Est-ce de la glace?
    ÉCRIRE "C'est de la glace"
FINSI
SI Temp > 0 ET Temp < 100 ALORS                  \\ Est-ce du liquide?
    ÉCRIRE "C'est du liquide"
FINSI
SI Temp >= 100 ALORS                              \\ Est-ce de la vapeur?
    ÉCRIRE "C'est de la vapeur"
FINSI

```

Pseudo-code 9-6 : Séquence de structures SI reliées

Notez que cet algorithme est assez laborieux. Les conditions se ressemblent plus ou moins, mais surtout on oblige le flux d'exécution à examiner trois conditions successives alors que toutes portent sur un même thème, soit la température (la valeur de la variable **Temp**). Il est cependant plus rationnel d'*imbriquer* les structures :

```

ÉCRIRE "Température de l'eau? "
LIRE Temp
SI Temp <= 0 ALORS                \\ Est-ce de la glace?
  ÉCRIRE "C'est de la glace"
SINON
  SI Temp > 0 ET Temp < 100 ALORS  \\ Est-ce du liquide?
    ÉCRIRE "C'est du liquide"
  SINON                            \\ C'est donc de la vapeur
    ÉCRIRE "C'est de la vapeur"
  FINSI
FINSI
FINSI

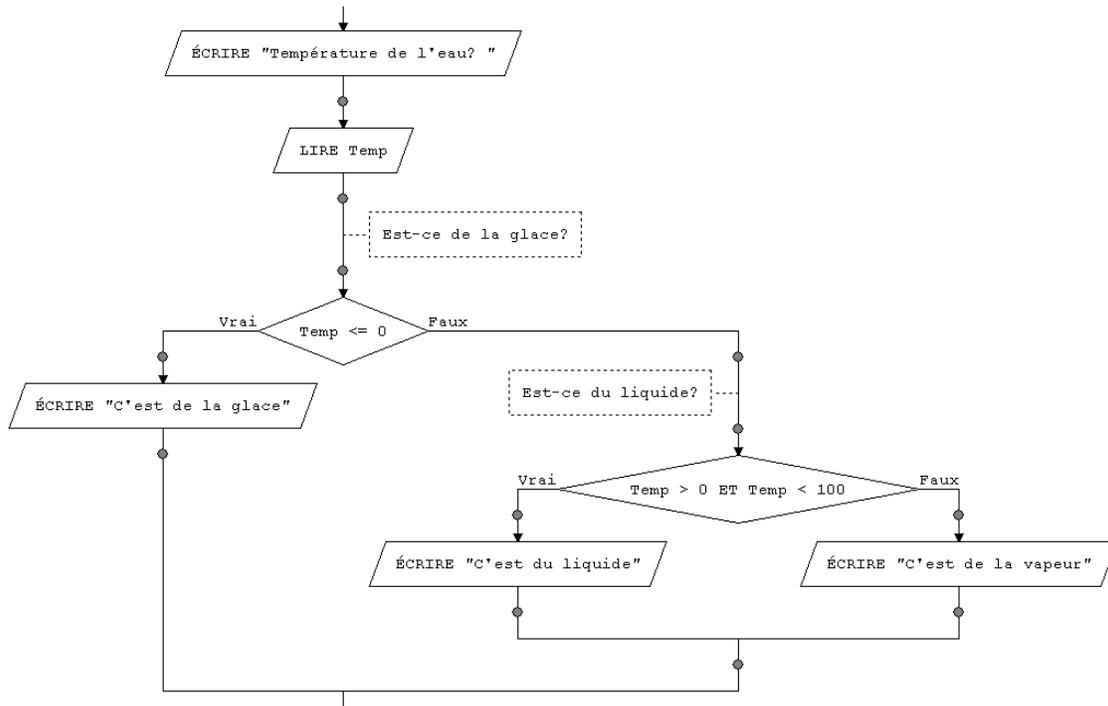
```

Pseudo-code 9-7 : Structures SI-SINON imbriquées les unes dans les autres

Nous avons fait des économies en termes de pseudo-code : au lieu de devoir taper trois conditions, nous n'avons plus que deux conditions. De plus, nous avons fait des économies en temps d'exécution de l'algorithme. En effet, si la température est inférieure à zéro, celui-ci écrit dorénavant « C'est de la glace » et le flux d'exécution passe directement après le dernier **FINSI**, sans examiner d'autres possibilités (qui sont forcément fausses).

Cette deuxième version est donc non seulement plus simple à écrire et plus lisible, mais également plus performante à l'exécution. Les structures conditionnelles imbriquées sont donc un outil indispensable à la simplification et à l'optimisation des algorithmes.

Notez que les structures conditionnelles peuvent aussi être imbriquées dans un organigramme :



Il est cependant important de souligner le principal danger relié à l'utilisation de structures conditionnelles imbriquées dans un pseudo-code : à chaque commande **SI...ALORS** doit correspondre un **FINSI**. Souvent, un **FINSI** manquant ou de trop sera facilement repéré si le pseudo-code est *indenté* adéquatement (i.e. décalé à droite à l'intérieur des structures, comme c'est le cas dans les pseudo-codes précédents).

9.8 Structure SI-SINON-SI

Comme nous l'avons vu dans la section précédente, l'emploi de [structures conditionnelles imbriquées](#) engendre une économie en temps d'exécution puisque le flux d'exécution quitte la structure dès qu'une [condition](#) est satisfaite et la séquence d'instructions correspondante est exécutée.

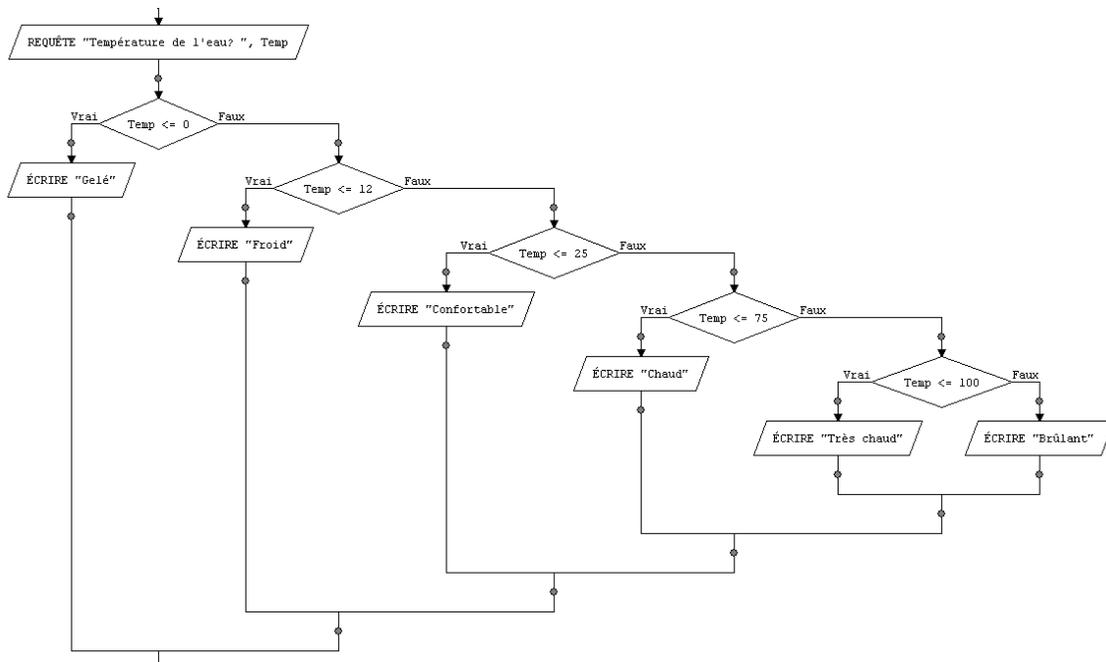
L'avantage des structures conditionnelles imbriquées est par contre amoindri par une complexité algorithmique plus grande lorsque plusieurs conditions sont impliquées. Le Pseudo-code 9-8 est un exemple de structures conditionnelles imbriquées rendant l'algorithme difficile à comprendre :

```
REQUÊTE "Température de l'eau? ", Temp
SI Temp <= 0 ALORS
  ÉCRIRE "C'est gelé"
SINON
  SI Temp <= 12 ALORS
    ÉCRIRE "C'est froid"
  SINON
    SI Temp <= 25 ALORS
      ÉCRIRE "C'est confortable"
    SINON
      SI Temp <= 75 ALORS
        ÉCRIRE "C'est chaud"
      SINON
        SI Temp <= 100 ALORS
          ÉCRIRE "C'est très chaud"
        SINON
          ÉCRIRE "C'est brûlant"
        FINSI
      FINSI
    FINSI
  FINSI
FINSI
```

Pseudo-code 9-8 : Structures conditionnelles imbriquées

On constate que ce pseudo-code peut porter à confusion et qu'il est facile d'oublier un **FINSI** lors de sa rédaction.

Les structures imbriquées nombreuses causent aussi des problèmes de compréhension de l'algorithme lorsque exprimées dans un organigramme. Dans l'Organigramme 9-4 la structure résultante est très large et ne peut pas être affichée en entier dans la fenêtre de l'[éditeur graphique](#) (à moins de [réduire le format d'affichage](#) à une taille telle que l'organigramme en devient généralement illisible à l'écran) :



Organigramme 9-4 : Structures conditionnelles imbriquées

La *structure conditionnelle SI-SINON-SI* vise à simplifier l'utilisation de structures imbriquées dans un contexte où le flux d'exécution doit quitter la structure dès qu'une condition est satisfaite et la séquence d'instructions correspondante exécutée :

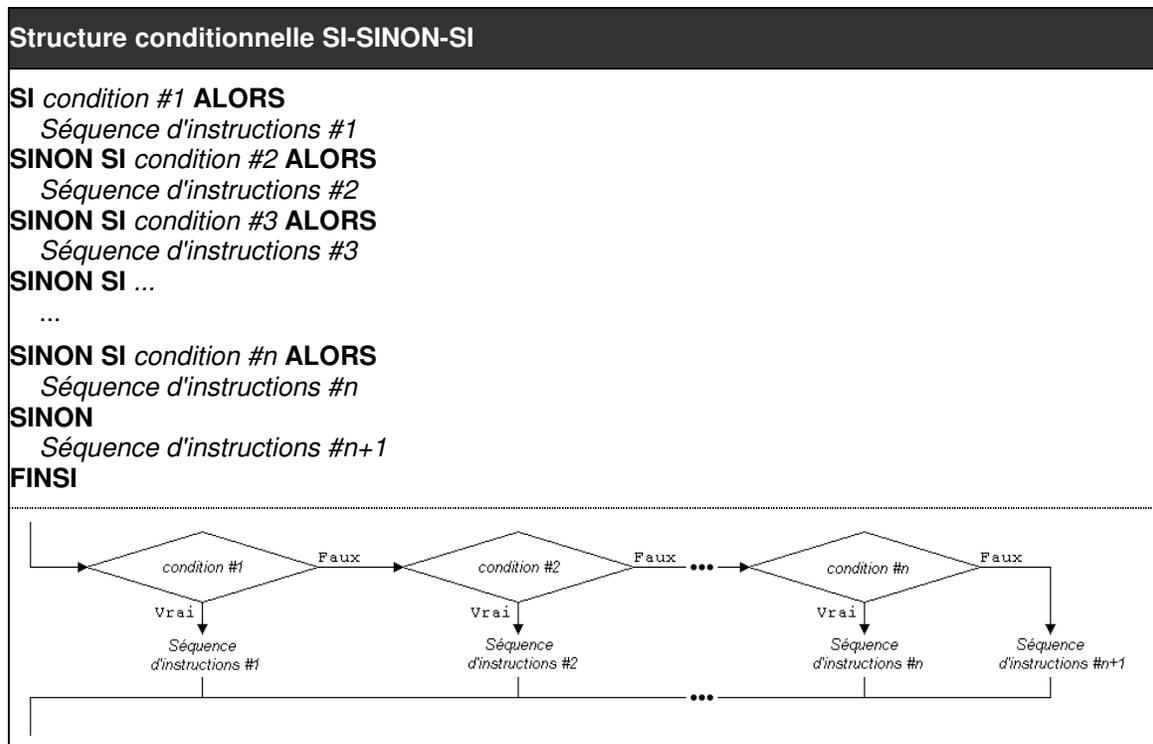


Tableau 9-6 : Structure conditionnelle SI-SINON-SI

Cette structure conditionnelle est employée lorsqu'on doit exécuter une et une seule séquence d'instructions en fonction d'une condition associée. Cette structure peut être interprétée ainsi :

- exécuter *Séquence d'instructions #1* si et seulement si *condition #1* est vraie;
- exécuter *Séquence d'instructions #2* si et seulement si *condition #1* est fausse et *condition #2* est vraie;
- exécuter *Séquence d'instructions #3* si et seulement si *condition #1* et *condition #2* sont fausses, mais *condition #3* est vraie;
- ...
- exécuter *Séquence d'instructions #i* si et seulement si *condition #1* à *condition #i-1* sont fausses, mais *condition #i* est vraie;
- finalement, si aucune des conditions de la structure n'est vraie et la structure dispose d'une section **SINON**, la *Séquence d'instructions #n+1* est exécutée.

La structure conditionnelle SI-SINON-SI en organigramme est construite en utilisant deux instructions d'organigrammes :

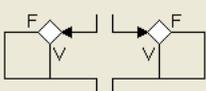
Instructions	Descriptions
	Structure conditionnelle <i>SI-SINON-SI</i> : structure conditionnelle comportant plusieurs séquences d'instructions alternatives dont une seule est exécutée en fonction du résultat de l'évaluation de conditions.
	Branchement pour structures conditionnelles : permettent d'insérer des branchements conditionnels supplémentaires dans les structures de sélection et les structures conditionnelles <i>SI-SINON-SI</i> .

Tableau 9-7 : Instructions d'organigrammes requises dans une structure SI-SINON-SI

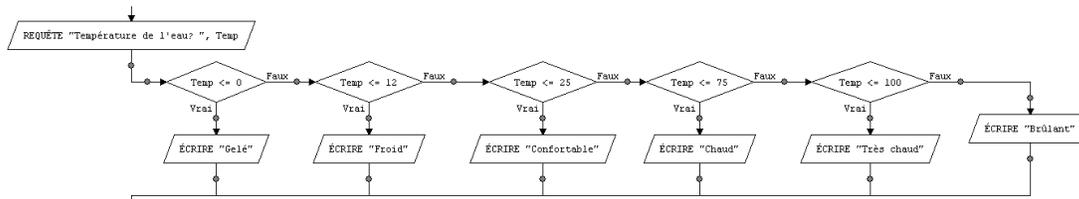
Voici l'exemple précédent reformulé à l'aide d'une structure conditionnelle SI-SINON-SI sous forme de pseudo-code et d'organigramme :

```

REQUÊTE "Température de l'eau? ", Temp
SI Temp <= 0 ALORS
    ÉCRIRE "C'est gelé"
SINON SI Temp <= 12 ALORS
    ÉCRIRE "C'est froid"
SINON SI Temp <= 25 ALORS
    ÉCRIRE "C'est confortable"
SINON SI Temp <= 75 ALORS
    ÉCRIRE "C'est chaud"
SINON SI Temp <= 100 ALORS
    ÉCRIRE "C'est très chaud"
SINON
    ÉCRIRE "C'est brûlant"
FINSI
    
```

Pseudo-code 9-9 : Structure conditionnelle SI-SINON-SI

Dans une structure SI-SINON-SI en pseudo-code, la dernière séquence d'instructions (**SINON Séquence d'instructions #n+1**) est optionnelle. Il en est de même dans la structure SI-SINON-SI en organigramme.



Organigramme 9-5 : Structure conditionnelle SI-SINON-SI

Alors que les deux formes de structures conditionnelles (les SI-SINON imbriquées et le SI-SINON-SI) sont équivalentes, la seconde est plus populaire parce qu'elle évite l'indentation profonde du code vers la droite. Une telle indentation ne laisse souvent que peu d'espace sur une ligne de pseudo-code et en force la continuation sur plus d'une ligne (à l'aide de \$), diminuant par conséquent la lisibilité de l'algorithme. La structure SI-SINON-SI en organigramme est aussi plus lisible que sa contrepartie imbriquée grâce à sa linéarité.

Comment *LARP* fait-il la distinction entre une structure conditionnelle SI-SINON-SI et des structures conditionnelles imbriquées dans un pseudo-code ? Lorsque les mots réservés **SINON** et **SI** se retrouvent de suite sur une même ligne, alors *LARP* suppose qu'ils font parti d'une structure SI-SINON-SI.

9.9 Structure de sélection

Un algorithme peut parfois contenir une série de décisions dans laquelle une variable ou une expression est testée séparément pour chacune des valeurs qui peut en résulter, et où différentes séquences d'instructions sont exécutées conséquemment. *LARP* offre la *structure de sélection* pour représenter de telles prises de décisions.

La *structure de sélection* permet de formuler une *structure SI-SINON-SI* tout en améliorant la lisibilité de l'algorithme. Considérez la section d'algorithme suivant, écrite en utilisant une structure SI-SINON-SI :

```

LIRE Valeur1, Valeur2, Opérateur
SI Opérateur = '+' ALORS
    Résultat = Valeur1 + Valeur2
SINON SI Opérateur = '-' ALORS
    Résultat = Valeur1 - Valeur2
SINON SI Opérateur = '*' OU Opérateur = 'x' ALORS
    Résultat = Valeur1 * Valeur2
SINON SI Opérateur = '/' ALORS
    Résultat = Valeur1 / Valeur2
SINON
    ÉCRIRE "Opérateur erroné"
FINSI
ÉCRIRE Résultat

```

Pseudo-code 9-10 : Structure SI-SINON-SI pour tester une même valeur

La formulation de ce pseudo-code en utilisant la structure de sélection (à l'aide des mots réservés **SÉLECTIONNER**, **SINON** et **FINSÉLECTIONNER**) donne :

```

LIRE Valeur1, Valeur2, Opérateur
SÉLECTIONNER Opérateur
    '+'      : Résultat = Valeur1 + Valeur2
    '-'      : Résultat = Valeur1 - Valeur2
    '*', 'x' : Résultat = Valeur1 * Valeur2
    '/'      : Résultat = Valeur1 / Valeur2
SINON
    Résultat = "Opérateur erroné"
FINSÉLECTIONNER
ÉCRIRE Résultat

```

Pseudo-code 9-11 : Structure de sélection

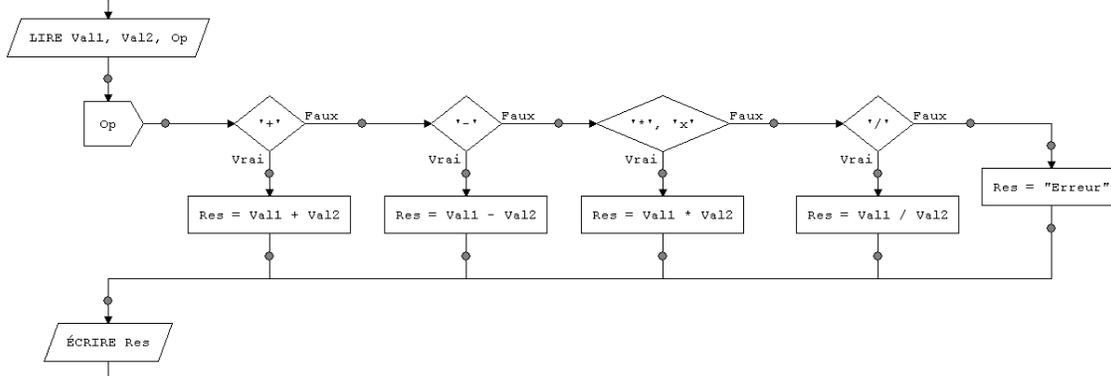
Dans l'exemple ci-dessus, la valeur de la variable **Opérateur** est comparée à chaque constante spécifiée. Si une constante testée correspond à la valeur de **Opérateur**, la séquence d'instructions associée à cette constante est exécutée.

Si la valeur de la variable **Opérateur** ne correspond pas à une constante, elle est comparée à la constante suivante, et ainsi de suite. La section **SINON** permet de prendre les mesures nécessaires lorsque la variable **Opérateur** ne correspond à aucune des constantes énumérées.

Notez que :

- l'expression testée doit être comparée à des **constantes** (ex: '+'),
- plus d'une constante peuvent être associées à une même séquence d'instructions (ex: '**', 'x'), et
- la section **SINON** est optionnelle.

Sous forme d'organigramme, la structure de sélection est similaire à la structure conditionnelle SI-SINON-SI, avec l'ajout d'un composant initial contenant l'expression à comparer aux constantes :



Organigramme 9-6 : Exemple de structure de sélection

Voici la syntaxe généralisée de la structure de sélection :

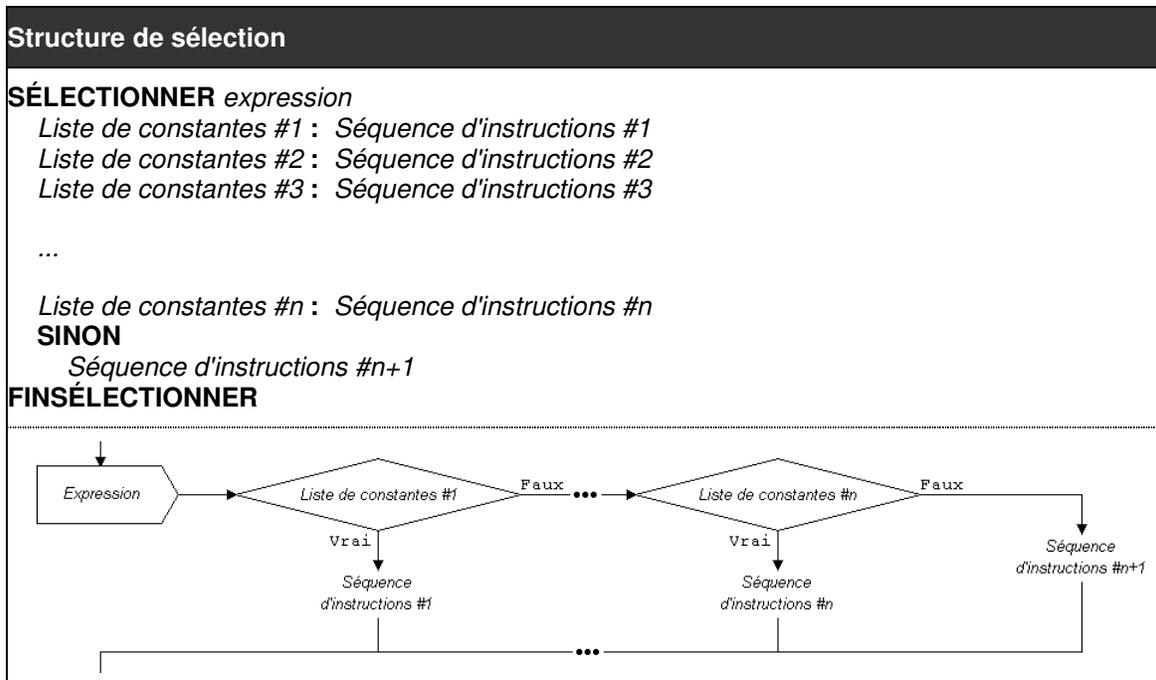


Tableau 9-8 : Structure de sélection

Voici les caractéristiques de cette structure :

- Une *Liste de constantes* peut être constituée d'une ou plusieurs constantes séparées par des virgules; ne pas oublier le symbole : à la fin de la liste dans le pseudo-code.
- Chaque *Séquence d'instructions* peut contenir une ou plusieurs instructions LARP. Dans un pseudo-code la première instruction peut être positionnée après le symbole :, sur la même ligne que la *Liste de constantes* correspondante.
- La section **SINON** et sa séquence d'instructions sont optionnelles. Dans un organigramme la *Séquence d'instructions #n+1* est aussi optionnelle.

Lors de l'exécution d'une structure de sélection, la valeur de l'*expression* est comparée successivement aux constantes retrouvées dans chaque *Liste de constantes*, en commençant par la première (*Liste de constantes #1*). Lorsqu'une constante correspondant à la valeur de l'*expression* est trouvée, la *Séquence d'instructions* correspondante est exécutée puis le flux d'exécution quitte la structure de sélection. Si aucune constante correspondant à la valeur de l'*expression* n'est trouvée et que la structure dispose d'une section **SINON**, la *Séquence d'instructions* correspondante (*Séquence d'instructions #n+1*) est exécutée.

Si plus d'une *Liste de constantes* contient la constante correspondant à la valeur de l'*expression*, seule la *Séquence d'instructions* associée à la première *Liste de constantes* est exécutée, puisque le flux d'exécution quitte la structure de sélection après l'exécution de cette séquence d'instructions.

Comme pour la structure SI-SINON-SI, la structure de sélection en organigramme est construite dans l'*éditeur graphique* en utilisant deux *instructions d'organigrammes* :

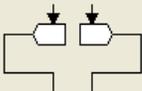
Instructions	Description
	<p>Structure de sélection : <i>structure conditionnelle</i> comportant plusieurs séquences d'instructions alternatives dont une seule est exécutée en fonction de la valeur d'une expression arithmétique.</p>
	<p>Branchement pour structures conditionnelles : permettent d'insérer des branchements conditionnels supplémentaires dans les <i>structures de sélection</i> et les <i>structures conditionnelles SI-SINON-SI</i>.</p>

Tableau 9-9 : Instructions d'organigrammes requises dans une structure de sélection

10 Structures répétitives

Il arrive fréquemment qu'un algorithme doive répéter une séquence d'instructions un certain nombre de fois afin d'accomplir une tâche; en fait, la plupart des algorithmes requièrent de telles répétitions. *LARP* offre trois structures à cette fin, généralement appelées *structures répétitives* ou *boucles* :

1. La *structure TANTQUE*
2. La *structure RÉPÉTER-JUSQU'À*
3. La *structure POUR*

Les structures répétitives sont basées sur l'évaluation d'une *condition*, dont le résultat est vrai ou faux. C'est sur la base de cette condition que la répétition est déterminée.

10.1 Structure TANTQUE

Dans sa forme la plus simple (*structure TANTQUE*) telle que présentée dans le Tableau 10-1, une structure répétitive sous forme pseudo-code est composée des mots réservés **TANTQUE**, **FAIRE** et **FINTANTQUE**, d'une *condition* et d'une *séquence d'instructions* à exécuter tant que la condition est vraie.

Dans un organigramme (voir Tableau 10-1), la structure TANTQUE comprend une condition dans un hexagone suivie de la *séquence d'instructions* sur la branche étiquetée *Vrai*. L'orientation de la branche *Faux* peut être à gauche ou à droite de la condition. Notez que la branche *Vrai* retourne à la condition, indiquant ainsi que le flux d'exécution retourne évaluer la condition une fois la *séquence d'instructions* exécutée.

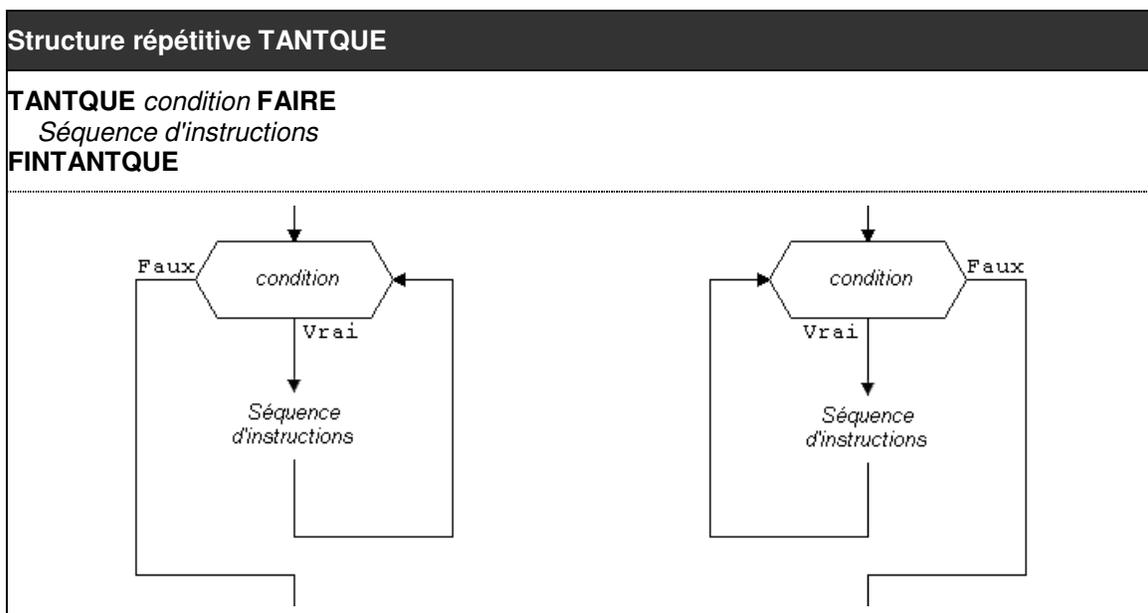


Tableau 10-1 : Structure répétitive TANTQUE

Dans la structure pseudo-code ci-dessus, le mot réservé **TANTQUE** indique le début de la structure répétitive, et le mot réservé **FINTANTQUE** en indique la fin. Dans la structure en

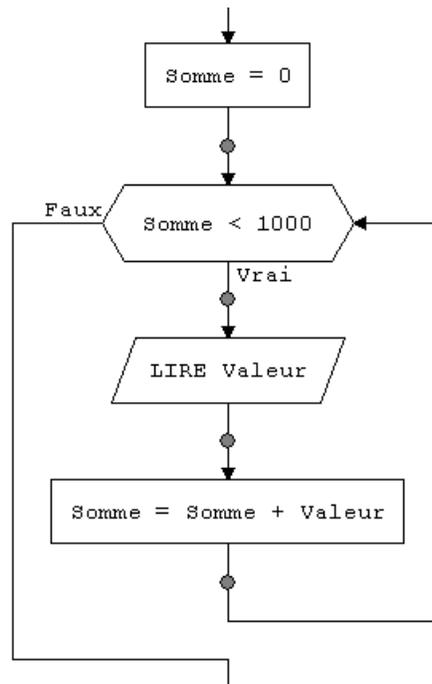
organigramme, la condition indique le début de la structure répétitive et la fin de la branche étiquetée *Faux* en indique la fin.

Les structures répétitives sont basées sur l'évaluation d'une condition, dont le résultat est vrai ou faux. C'est sur la base de cette condition que le flux d'exécution est déterminé. Dans le cas d'une structure TANTQUE, la *séquence d'instructions* est exécutée tant que la *condition* est satisfaite (i.e. vraie).

L'exemple ci-dessous (sous forme pseudo-code et d'organigramme) exploite une structure TANTQUE afin d'additionner des valeurs lues jusqu'à ce que leur somme atteigne ou dépasse **1000** :

```
Somme = 0
TANTQUE Somme < 1000 FAIRE
  LIRE Valeur
  Somme = Somme + Valeur
FINTANTQUE
```

Pseudo-code 10-1 : Structure répétitive TANTQUE



Organigramme 10-1 : Structure répétitive TANTQUE

La condition d'une structure TANTQUE est évaluée avant chaque *itération* (une itération est une et une seule exécution de la *séquence d'instructions* dans la boucle). Ainsi, l'exécution de la structure peut être résumée ainsi :

1. vérifier la *condition*
2. si elle est vraie alors
 - 2.1. exécuter la *séquence d'instructions*
 - 2.2. retourner à l'étape 1.

Ainsi, la condition doit obligatoirement être satisfaite pour que la séquence d'instructions soit exécutée. Dès que la condition devient non satisfaite (i.e. fausse), le flux d'exécution est redirigé vers les instructions suivant la structure répétitive (après le **FINTANTQUE** en pseudo-code).

Une particularité de la structure TANTQUE est que la *séquence d'instructions* peut ne pas être exécutée du tout si la première évaluation de la condition (i.e. lorsque le flux d'exécution entre dans la structure répétitive) retourne faux. En effet, si la condition est fausse dès le départ, aucune itération de la boucle ne sera effectuée.

Puisque qu'une exécution de la *séquence d'instructions* dans une structure répétitive est généralement appelée une *itération*, les structures répétitives sont aussi appelées *structures itératives*. Ce sont des synonymes.

10.2 Structure RÉPÉTER-JUSQU'À

La *structure RÉPÉTER-JUSQU'À* est semblable à la structure TANTQUE. Comme cette dernière, elle inclut l'exécution d'une *séquence d'instructions* à répétition et en fonction de la valeur d'une *condition*. Cependant, les structures RÉPÉTER-JUSQU'À et TANTQUE diffèrent sur deux points :

1. La structure TANTQUE exécute la séquence d'instructions tant et aussi longtemps que la condition est satisfaite, alors que la structure RÉPÉTER-JUSQU'À exécute la séquence d'instructions tant et aussi longtemps que la condition *n'est pas* satisfaite. En d'autres mots, la structure RÉPÉTER-JUSQU'À itère *jusqu'à* ce que la condition devienne vraie.
2. La structure TANTQUE vérifie la condition *avant* chaque *itération*, alors que la structure RÉPÉTER-JUSQU'À vérifie la condition *après* chaque itération.

La différence principale entre ces deux structures réside dans le fait que, pour la structure RÉPÉTER-JUSQU'À, la séquence d'instructions est exécutée *au moins une fois*, sans égard à la valeur de la condition. Cette caractéristique est mise en évidence par la position de la condition dans la structure : elle est située à la fin de celle-ci (alors que dans la structure TANTQUE la condition est située au début de la structure).

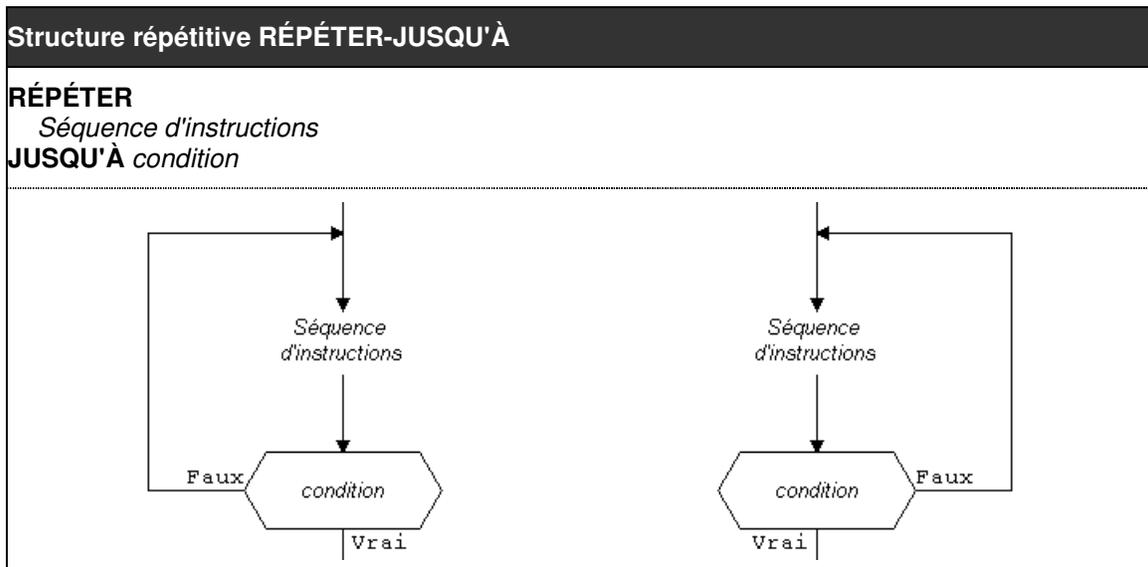


Tableau 10-2 : Structure répétitive RÉPÉTER-JUSQU'À

La structure RÉPÉTER-JUSQU'À sous forme pseudo-code, présentée dans le Tableau 10-2, est composée des mots réservés **RÉPÉTER** et **JUSQU'À**, d'une condition et d'une *séquence*

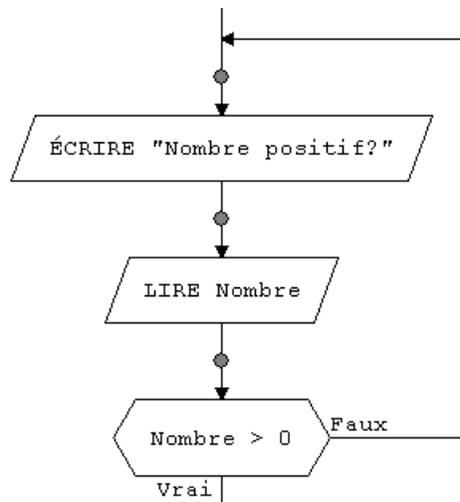
d'instructions à exécuter jusqu'à ce que la condition devienne vraie (en d'autres mots, tant qu'elle est fausse). Dans un organigramme la structure RÉPÉTER-JUSQU'À comprend une condition dans un hexagone précédée de la *séquence d'instructions*. L'orientation de la branche *Faux*, qui représente l'itération, peut être à gauche ou à droite de la condition. Notez que la branche *Vrai* sort de la boucle, indiquant ainsi que le flux d'exécution cesse d'exécuter la *séquence d'instructions* lorsque la condition est vraie.

L'exemple ci-dessous exploite une structure RÉPÉTER-JUSQU'À afin de lire un entier positif et le valider :

```

RÉPÉTER
  ÉCRIRE "Nombre positif?"
  LIRE Nombre
JUSQU'À Nombre > 0
  
```

Pseudo-code 10-2 : Structure répétitive RÉPÉTER-JUSQU'À



Organigramme 10-2 : Structure répétitive RÉPÉTER-JUSQU'À

Puisque au moins une lecture doit être effectuée, la structure RÉPÉTER-JUSQU'À est préférable car elle fait obligatoirement une itération (i.e. une lecture) avant de vérifier la condition. La structure RÉPÉTER-JUSQU'À est généralement préférée à la structure TANTQUE lorsque les variables dont dépend la condition reçoivent leur valeur dans la séquence d'instructions, ce qui exige obligatoirement une première itération.

Notez cependant qu'une structure TANTQUE peut remplacer toute structure RÉPÉTER-JUSQU'À, au coût de quelques instructions supplémentaires (dans le Pseudo-code 10-3 une opération de lecture supplémentaire doit être insérée avant la première itération) :

```

ÉCRIRE "Nombre positif?"
LIRE Nombre
TANTQUE Nombre <= 0 FAIRE
  ÉCRIRE "Nombre positif?"
  LIRE Nombre
FINTANTQUE
  
```

Pseudo-code 10-3 : Structure TANTQUE équivalente à la boucle du Pseudo-code 10-2

10.3 Structure POUR

La troisième structure répétitive est plus sophistiquée que les deux premières (les structures **TANTQUE** et **RÉPÉTER-JUSQU'À**). La *structure POUR* a la forme suivante :

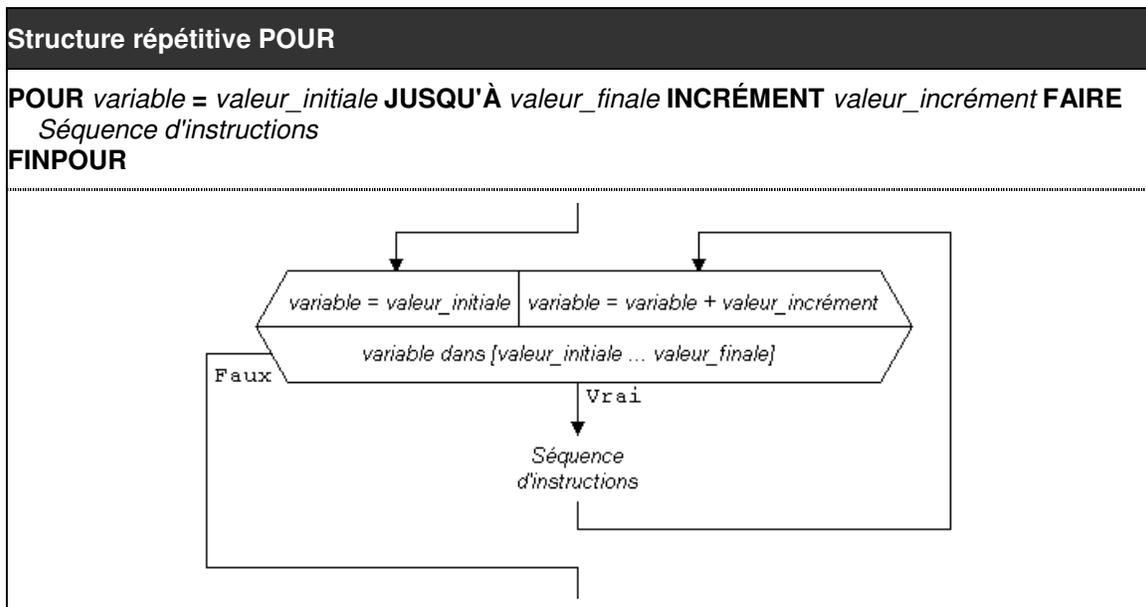


Tableau 10-3 : Structure répétitive POUR

Dans la structure pseudo-code ci-dessus, le mot réservé **POUR** indique le début de la structure répétitive, et le mot réservé **FINPOUR** en indique la fin.

La structure répétitive POUR est généralement employée lorsqu'on veut répétitivement faire varier la valeur d'une variable (identificateur *variable* ci-dessus) d'une valeur initiale (identificateur *valeur_initiale* ci-dessus) jusqu'à une valeur finale (identificateur *valeur_finale* ci-dessus), tout en exécutant une *séquence d'instructions* pour chaque valeur de cette variable. L'identificateur *valeur_incrément* indique le changement à appliquer à *variable* à la fin de chaque itération.

Ainsi, dans la structure ci-dessus, les identificateurs signifient :

- *variable* : variable dont la valeur varie de *valeur_initiale* à *valeur_finale*, changeant de *valeur_incrément* unités à chaque itération.
- *valeur_initiale* : valeur de la variable à la première itération.
- *valeur_finale* : valeur de la variable à la dernière itération.
- *valeur_incrément* : incrément appliqué à la variable après chaque itération.
- *Séquence d'instructions* : instructions exécutées à chaque itération. La *variable* peut être employée dans ces instructions.

Notez que l'incrément peut être omis de la structure POUR. Dans ce cas, l'incrément par défaut appliqué est de 1. La *variable* employée pour compter les itérations dans une structure POUR est généralement appelée une *variable d'itération*.

Voici un exemple d'utilisation d'une telle structure. Supposons qu'on veuille afficher les températures en Fahrenheit correspondant aux 20 premiers degrés Celsius du thermomètre. On peut produire une telle grille de températures avec une structure **TANTQUE** :

```

\\ Affiche les températures de 0C à 19C en Fahrenheit
Celsius = 0
TANTQUE Celsius <= 19 FAIRE
    Fahrenheit = Celsius * 9/5 + 32
    ÉCRIRE Celsius, ' = ', Fahrenheit
    Celsius = Celsius + 1
FINTANTQUE

```

Pseudo-code 10-4 : Itérer d'une valeur à la suivante avec TANTQUE

La structure ci-dessus effectue exactement 20 itérations, augmentant de 1 la valeur de la variable **Celsius** à chaque itération.

La structure POUR offre une syntaxe plus compacte et naturelle pour représenter les structures répétitives telles que celle ci-dessus :

```

\\ Affiche les températures de 0C à 19C en Fahrenheit
POUR Celsius = 0 JUSQU'À 19 FAIRE
    Fahrenheit = Celsius * 9/5 + 32
    ÉCRIRE Celsius, ' = ', Fahrenheit
FINPOUR

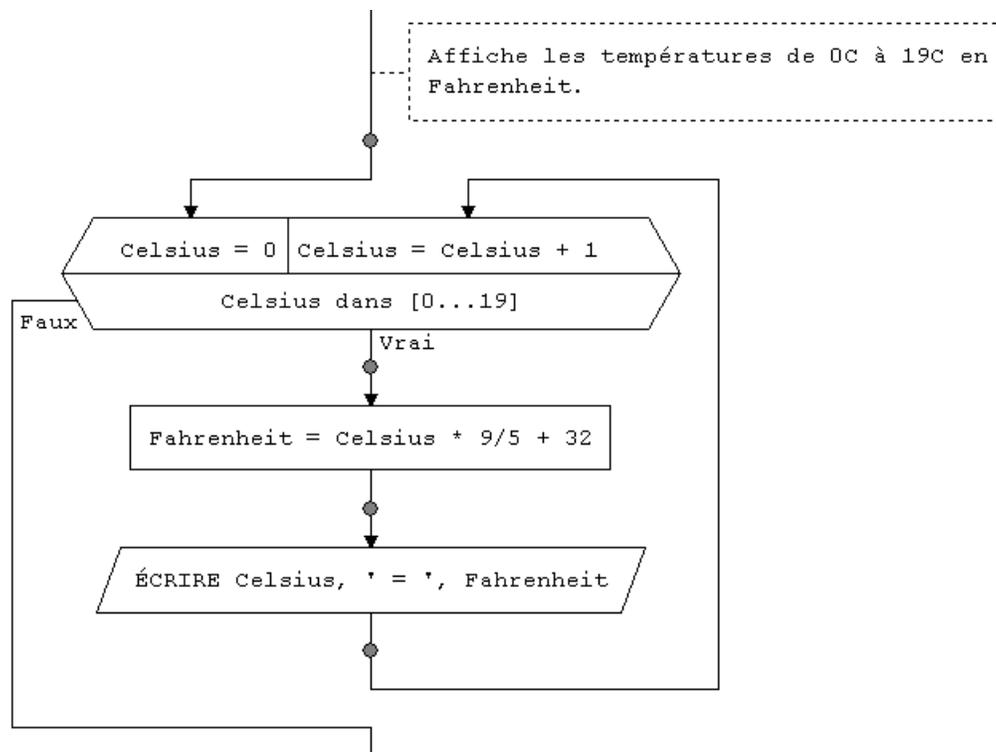
```

Pseudo-code 10-5 : Structure répétitive POUR

Dans cette structure, la variable **Celsius** est initialisée à **0** avant la première itération, puis successivement augmentée de **1** à la fin de chaque itération (1 est l'incrément par défaut puisque aucune valeur d'incrément n'est fournie dans la structure POUR). On constate donc que l'incrément est implicite (nul besoin de l'incrémenter explicitement dans la boucle avec l'instruction **Celsius = Celsius + 1**). Lorsque la variable **Celsius** atteint la valeur **20**, le flux d'exécution quitte la structure POUR et poursuit l'exécution du pseudo-code suivant la boucle, après le **FINPOUR**.

L'Organigramme 10-3 présente le même algorithme que Pseudo-code 10-5. Notez que la structure comporte une condition sous une forme différente de celles retrouvées dans les structures TANTQUE et RÉPÉTER-JUSQU'À en organigramme. Le symbole hexagonal de la structure POUR contient tous les éléments de son équivalente en pseudo-code : l'initialisation de la variable d'itération en entrée de la structure (**Celsius = 0**), une vérification de continuer les itérations (**Celsius dans [0...19]**) et l'incrément de la variable d'itération à la fin de chaque itération (**Celsius = Celsius + 1**). Les branchements de la structure POUR en organigramme indiquent clairement le parcours du flux d'exécution dans la structure :

1. En entrant dans la structure, l'instruction d'initialisation (**Celsius = 0**) est exécutée une et une seule fois.
2. La variable d'itération est ensuite validée en fonction des valeurs limites d'itération (**Celsius dans [0...19]**). Si la valeur de la variable d'itération est dans cette intervalle, alors il y a itération :
 - 2.1. Les deux instructions dans la boucle sont exécutées.
 - 2.2. La variable d'itération est incrémentée (**Celsius = Celsius + 1**).
 - 2.3. Enfin le flux d'exécution retourne à l'étape 2 afin de valider la valeur de la variable d'itération.



Organigramme 10-3 : Structure répétitive POUR

L'option de spécifier une valeur incrément dans la structure POUR offre la possibilité d'utiliser un incrément autre que 1 dans une boucle. Ainsi, l'exemple suivant convertit seulement les températures Celsius paires en Fahrenheit :

```

\\ Affiche les températures 0C, 2C, 4C, 6C... à 18C en Fahrenheit
POUR Celsius = 0 JUSQU'À 18 INCRÉMENT 2 FAIRE
    Fahrenheit = Celsius * 9/5 + 32
    ÉCRIRE Celsius, ' = ', Fahrenheit
FINPOUR
  
```

Pseudo-code 10-6 : Structure POUR avec incrément autre que 1

Notez qu'une structure POUR incrémente la valeur de la variable d'une unité par défaut à chaque itération. Si cependant la *valeur_initiale* est supérieure à la *valeur_finale*, la variable sera automatiquement diminuée de un à chaque itération :

```

\\ Affiche les températures de 19C à 0C en Fahrenheit
POUR Celsius = 19 JUSQU'À 0 FAIRE
    Fahrenheit = Celsius * 9/5 + 32
    ÉCRIRE Celsius, ' = ', Fahrenheit
FINPOUR
  
```

Pseudo-code 10-7 : Structure répétitive POUR avec décrémentation

On peut aussi spécifier une valeur d'incrément négative afin de réduire la variable (**Celsius**) par bonds autres que d'une seule unité :

```

\\ Affiche les températures de 18C, 16C, 14C ... à 0C en Fahrenheit
POUR Celsius = 18 JUSQU'À 0 INCRÉMENT -2 FAIRE
    Fahrenheit = Celsius * 9/5 + 32
    ÉCRIRE Celsius, ' = ', Fahrenheit
FINPOUR

```

Pseudo-code 10-8 : Structure répétitive POUR avec décrétement autre que -1

La valeur de la variable d'itération (**Celsius** dans l'exemple ci-dessus) peut être employée dans les instructions incluses dans la structure POUR, mais elle ne peut pas être modifiée par ces instructions. Ainsi, dans Pseudo-code 10-9, l'instruction **LIRE i** est interdite car elle vise à modifier la valeur de la variable d'itération *i*. Par contre, l'instruction **Log(i * 100)** est permise dans la boucle, tout comme **Fahrenheit = Celsius * 9/5 + 32** dans Pseudo-code 10-8, puisqu'elles ne modifient pas la valeur de la variable d'itération.

```

POUR i = 0 JUSQU'À 10 FAIRE
    ÉCRIRE Log(i * 100)
    LIRE i
FINPOUR

```

Pseudo-code 10-9 : Structure POUR invalide

Même si elle s'avère souvent pratique, la structure POUR n'est pas indispensable; on pourrait fort bien programmer toutes les situations de boucle uniquement avec la structure TANTQUE. Le seul intérêt de la structure POUR est d'épargner un peu de travail au programmeur, en lui évitant de gérer lui-même la progression de la variable d'itération. Autrement dit, la structure POUR est un cas particulier de la structure TANTQUE : celui où le programmeur peut déterminer à l'avance le nombre d'itérations à effectuer.

Conceptuellement, on dit que la structure POUR est une *boucle inconditionnelle* car le nombre d'itérations effectuées par la boucle est prédéterminée et ne dépend pas de la séquence d'instructions dans la boucle. Par exemple dans le Pseudo-code 10-5 on connaît le nombre d'itérations qui seront effectuées par la boucle (20 itérations) avant même de débiter la première itération. Il en est de même dans Pseudo-code 10-6 (10 itérations) et dans Pseudo-code 10-7 (20 itérations). Les structures TANTQUE et RÉPÉTER-JUSQU'À sont généralement catégorisées comme *boucles conditionnelles* car le nombre d'itérations qu'elles effectuent dépend d'une condition dont la valeur est généralement déterminée par la séquence d'instructions dans la boucle. C'est par exemple le cas dans Pseudo-code 10-2 et Pseudo-code 10-3, où la variable **Nombre** est modifiée dans la boucle (ce n'est donc pas une variable d'itération).

La plupart des langages de programmation traditionnels offre une structure inconditionnelle équivalente à la structure POUR de LARP. Ainsi, *C++* et *Java* ont la boucle **for**, dont la syntaxe est similaire à POUR quoique plus complexe.

11 Modules

La *programmation modulaire* est une technique utilisée lors de la conception d'algorithmes complexes. Elle consiste à diviser un algorithme en sections. Chaque section est appelée un *module* et exécute une tâche simple.

Voici des exemples de tâches qu'un module peut exécuter :

- Afficher un menu d'options
- Afficher des résultats
- Calculer une moyenne de données
- Valider des données
- Trier des données

Un module est identifié par un *nom* unique et consiste en une séquence d'instructions débutant avec le mot réservé **ENTRER** et se terminant avec le mot réservé **RETOURNER**. La séquence d'instructions contenue dans le module est exécutée lorsque le nom du module est rencontré durant l'exécution d'autres modules. On dit alors que le module est *invoqué*.

LARP supporte trois types de modules :

- Les **modules simples** sont non paramétrés, n'acceptant aucun argument lors de leur invocation.
- Les **modules paramétrés** acceptent des arguments lors de l'invocation, permettant ainsi de contrôler leur exécution.
- Les **modules avec valeur de retour**, lorsqu'ils sont invoqués, retournent un résultat au module invoquant. Comme les modules paramétrés, ils peuvent accepter des arguments.

Les modules aident à structurer les algorithmes et permettent le développement de sections d'algorithme configurables et réutilisables. Les modules sont essentiellement des « fragments d'algorithmes ».

La gestion des modules d'un projet *LARP* est effectuée via le [navigateur de documents](#) ainsi que la [barre de menu](#). Ceux-ci permettent de créer et/ou supprimer les modules du projet.

11.1 Noms de modules

Les modules d'un projet *LARP* doivent être nommés selon les règles suivantes :

- Un nom de module doit débuter par une lettre (**A** à **Z**, **a** à **z**) ou le caractère de soulignement (**_**).
- Un nom de module peut être constitué de lettres minuscules, de lettres majuscules, de chiffres et du caractère de soulignement.
- Un nom de module ne doit pas correspondre à un mot réservé de *LARP*, tels que **ÉCRIRE**, **FIN**, **SI** et **PI**, et ce sans égard aux accents (par exemple **ECRIRE** est aussi un mot réservé).

- Le langage ne fait pas de distinction entre les lettres minuscules et les lettres majuscules, ce qui signifie que **Module_1**, **module_1** et **MODULE_1** font référence au même module.
- Le langage ignore les accents, ce qui signifie que **Coût**, **Cout** et **Coüt** font référence au même module.

Un module peut être ajouté au projet :

- via la **barre de menu**, en sélectionnant la commande **Fichier » Nouveau...**, puis en sélectionnant le type de module auxiliaire,
- via le bouton  du **panneau de contrôle**, ou
- via le menu contextuel du **navigateur de document**.

Avant d'ajouter un nouveau module au projet, *LARP* affiche la fenêtre suivante, où l'utilisateur doit spécifier le type de module auxiliaire à créer (pseudo-code ou organigramme) ainsi que son nom :

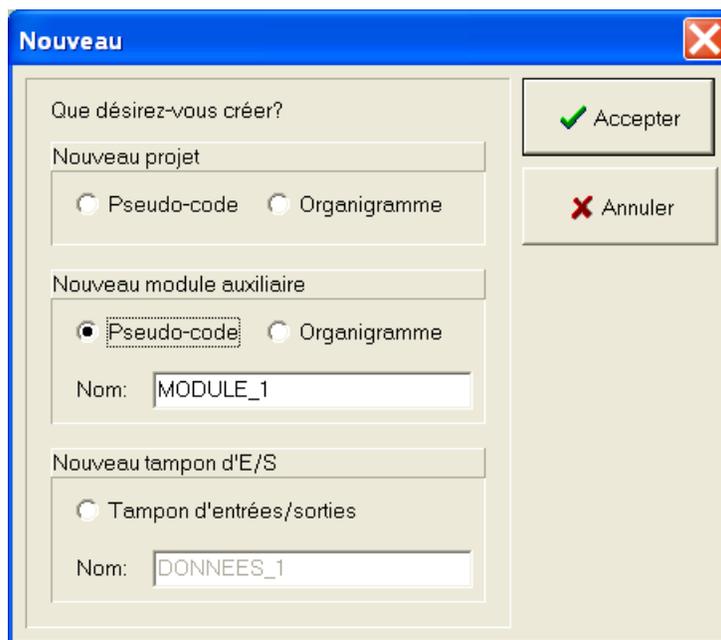


Figure 11-1 : Création d'un module auxiliaire

À noter que tous les modules d'un même projet doivent avoir un nom distinct.

11.2 Module principal

Lorsqu'un algorithme est divisé en plusieurs modules, un de ces modules doit tenir le rôle de *module principal*. Contrairement aux autres modules (appelés **modules auxiliaires**) qui débutent par le mot réservé **ENTRER** et se terminent par **RETOURNER**, le module principal débute par le mot réservé **DÉBUT** et se termine par le mot réservé **FIN** :

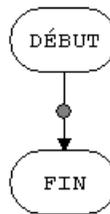
```

\\ Module principal
DÉBUT
  ÉCRIRE "Bonjour"
FIN

```

Pseudo-code 11-1 : Le module principal

Un projet *LARP* doit obligatoirement disposer d'un et un seul module principal, car *LARP* débute l'exécution de l'algorithme à l'instruction **DÉBUT** et termine l'exécution de l'algorithme à l'instruction **FIN**. Seul le module principal peut contenir les instructions **DÉBUT** et **FIN**. Par contre, un projet peut contenir ou non des modules auxiliaires, et ceux-ci doivent débiter avec l'instruction **ENTRER** et se terminer avec l'instruction **RETOURNER**.



Organigramme 11-1 : Le module principal

Lors de la création d'un nouveau projet, *LARP* crée automatiquement le module principal (nommé *PRINCIPAL*) de l'algorithme et y insère les instructions **DÉBUT** et **FIN**.

11.3 Modules auxiliaires

Les *modules simples* (i.e. sans paramètre) sont utilisés pour exécuter des tâches simples comme afficher des menus pour l'utilisateur. Le module se compose d'une séquence d'instructions regroupées entre les mots réservés **ENTRER** et **RETOURNER**.

Tous les modules autres que le module principal sont appelés *modules auxiliaires* et effectuent généralement des tâches requises par le module principal.

Voici un exemple de module auxiliaire affichant un menu :

```

\\ Module auxiliaire Menu
ENTRER
  ÉCRIRE "Le menu est"
  ÉCRIRE " 1 - Lire le dossier"
  ÉCRIRE " 2 - Sauvegarder le dossier"
  ÉCRIRE " 3 - Afficher le dossier"
  ÉCRIRE " 4 - Modifier le dossier"
  ÉCRIRE " 5 - Quitter"
RETOURNER

```

Pseudo-code 11-2 : Module auxiliaire

Le module ci-dessus, appelé **Menu**, exécute les instructions séquentiellement, de **ENTRER** jusqu'à **RETOURNER**. Pour l'invoquer dans un algorithme, le nom du module doit être spécifié dans un module (généralement un autre module du projet), précédé du mot réservé **EXÉCUTER** :

```

\\ Module principal
DÉBUT
  RÉPÉTER          \\ Afficher le menu
    EXÉCUTER Menu
    REQUÊTE "Commande? ", Commande
  JUSQU'À Commande = 5
FIN

```

Pseudo-code 11-3 : Invocation d'un module auxiliaire

Dans le module principal ci-dessus, le module simple **Menu** est exécuté (i.e. invoqué) à chaque itération afin d'afficher le menu. Toutes les instructions du module **Menu** sont exécutées à chaque invocation. L'instruction **REQUÊTE** suivant l'invocation du module est exécutée après chaque invocation de **Menu**.

Le résultat à la [console d'exécution](#) du module principal avec les valeurs 1 et 5 fournies par l'utilisateur est :

```

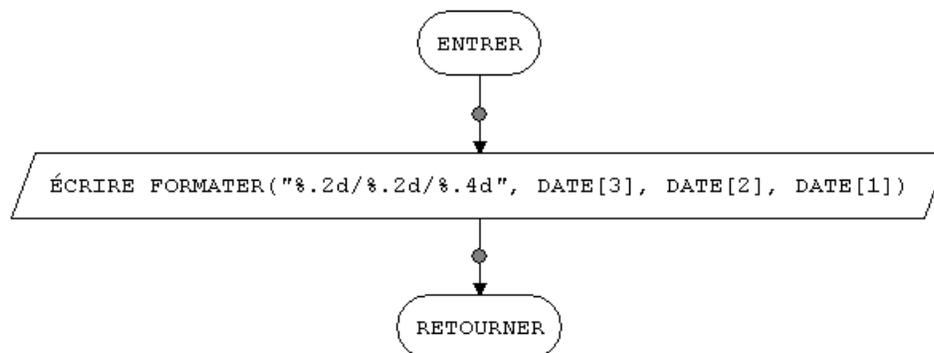
Le menu est
1 - Lire le dossier
2 - Sauvegarder le dossier
3 - Afficher le dossier
4 - Modifier le dossier
5 - Quitter
Commande? 1
Le menu est
1 - Lire le dossier
2 - Sauvegarder le dossier
3 - Afficher le dossier
4 - Modifier le dossier
5 - Quitter
Commande? 5

```

Figure 11-2 : Affichage lors de l'exécution du module auxiliaire **Menu**

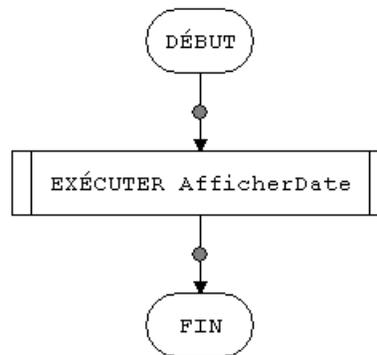
Évidemment, le module principal ci-dessus est incomplet puisque aucune action n'est entreprise lorsque l'utilisateur sélectionne les commandes 1 à 4.

Dans *LARP*, un module auxiliaire peut aussi être formulé sous forme d'organigramme. Le module auxiliaire suivant affiche la date courante sous la forme *JJ/MM/AAAA* :



Organigramme 11-2 : Module auxiliaire simple sous forme d'organigramme

Le module auxiliaire dans l'Organigramme 11-2, nommé **AfficherDate**, peut aussi être invoqué d'un module principal sous forme d'organigramme. Notez que *LARP* dispose d'une instruction d'organigramme spécifique à l'invocation de modules auxiliaires :



Organigramme 11-3 : Organigramme invoquant un module auxiliaire

11.4 Variables locales

Un module peut exploiter ses propres **variables** pour accomplir des tâches. Ces variables, appartenant exclusivement au module et n'étant pas partagées avec les autres modules du projet, sont appelées des *variables locales* car elles sont locales au module.

Les variables locales sont accessibles partout entre les mots réservés **ENTRER** et **RETOURNER** du **module auxiliaire** ou **DÉBUT** et **FIN** du **module principal**. Lorsque deux modules utilisent le même nom pour une variable locale, les deux variables sont distinctes. Les modules qui suivent illustrent cette indépendance :

<pre> \\ Module principal DÉBUT Valeur = 1 EXÉCUTER Module_A ÉCRIRE Valeur \\ Affiche 1 comme valeur FIN </pre>
<pre> \\ Module auxiliaire Module_A ENTRER Valeur = 2 RETOURNER </pre>

Pseudo-code 11-4 : Variables locales

Même si les deux modules exploitent une variable nommée **Valeur**, ces deux variables sont distinctes, celle du module principal n'étant pas modifiée par l'invocation du module auxiliaire.

La seule façon que deux modules puissent partager une donnée est via l'utilisation de **paramètres**.

11.5 Modules auxiliaires avec paramètres

Les **modules auxiliaires** peuvent accepter des valeurs, appelées *paramètres*, leur étant transmises lors de l'invocation. Les paramètres permettent ainsi au module *invoquant* (i.e. module contenant l'instruction **EXÉCUTER**) de fournir des données au module invoqué (i.e. le module visé par l'instruction **EXÉCUTER**).

Les paramètres d'un module auxiliaire permettent au module invoquant de « configurer » l'exécution du module invoqué en fonction d'une ou plusieurs valeurs spécifiées.

LARP offre deux types de paramètres de module :

1. **Paramètres valeurs** : le module invoquant peut transmettre une valeur au module invoqué via ce type de paramètres, mais le module invoqué ne peut pas transmettre un résultat via ce même paramètre. Par défaut, les paramètres d'un module sont de type valeur.
2. **Paramètres références** : le module invoquant peut transmettre une valeur au module invoqué via ce type de paramètres, et le module invoqué peut transmettre un résultat via ce même paramètre. Le paramètre doit être identifié comme référence avec le mot réservé **RÉFÉRENCE**.

La figure suivante illustre le transfert de données via chaque type de paramètres entre deux modules, un (*Module_A*) invoquant l'autre (*Module_B*) :

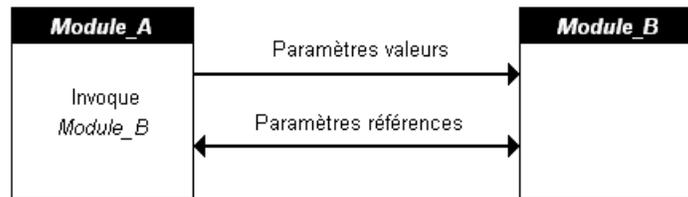


Figure 11-3 : Transfert de données via paramètres

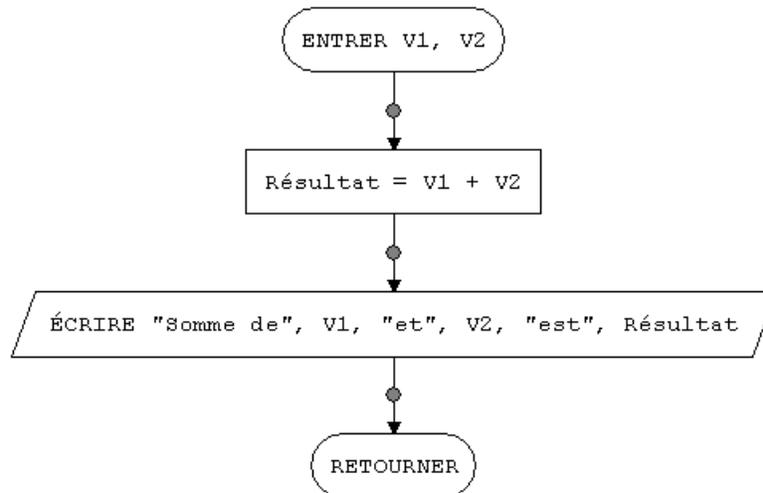
11.5.1 Déclaration des paramètres d'un module

Les paramètres d'un **module auxiliaire** sont des **variables** énumérées à droite du mot réservé **ENTRER**. Si plus d'un paramètre sont spécifiés, ceux-ci sont séparés par des virgules. Dans l'exemple suivant (Pseudo-code 11-5 et Organigramme 11-4) **V1** et **V2** sont les paramètres du module **Addition** :

```

\\ Module auxiliaire Addition
ENTRER V1, V2
  Résultat = V1 + V2
  ÉCRIRE "Somme de", V1, "et", V2, "est", Résultat
RETOURNER
  
```

Pseudo-code 11-5 : Module auxiliaire avec paramètres



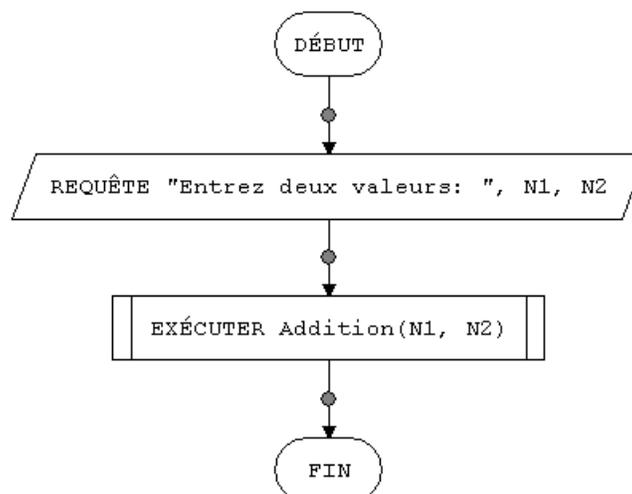
Organigramme 11-4 : Module auxiliaire avec paramètres

Les paramètres d'un module auxiliaire sont des variables dont le rôle est de recevoir des valeurs fournies lors de l'invocation. Les valeurs fournies lors de l'invocation sont appelées *arguments*, et doivent être énumérées entre parenthèses suivant l'invocation.

```

\\ Module principal
DÉBUT
  REQUÊTE "Entrez deux valeurs: ", N1, N2
  EXÉCUTER Addition(N1, N2)
FIN
  
```

Pseudo-code 11-6 : Invocation d'un module auxiliaire avec arguments



Organigramme 11-5 : Invocation d'un module auxiliaire avec arguments

Lors de l'invocation dans Pseudo-code 11-6 et Organigramme 11-5, les valeurs des arguments (**N1** et **N2**) sont respectivement copiées dans les paramètres du module invoqué (**V1** et **V2**). Ainsi, la valeur de **N1** est copiée dans **V1** et la valeur de **N2** est copiée dans **V2**. Puisque les

paramètres servent à recevoir les valeurs provenant des arguments, les paramètres doivent obligatoirement être des variables.

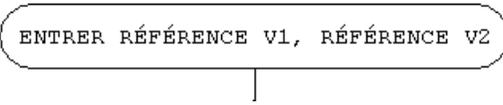
Par défaut, les paramètres énumérés dans l'instruction **ENTRER** d'un module sont dit **paramètres valeurs** car ils reçoivent les valeurs d'arguments correspondant lors d'une invocation. Un paramètre peut alternativement être désigné comme **paramètre référence** en précédant son nom dans l'instruction **ENTRER** du mot réservé **RÉFÉRENCE** :

```

\\ Module auxiliaire Interchanger
ENTRER RÉFÉRENCE V1, RÉFÉRENCE V2
    Temp = V1
    V1    = V2
    V2    = Temp
RETOURNER

```

Pseudo-code 11-7 : Déclaration de paramètres références



```

ENTRER RÉFÉRENCE V1, RÉFÉRENCE V2

```

Organigramme 11-6 : Déclaration de paramètres références

L'indicateur **RÉFÉRENCE** s'applique seulement au paramètre suivant le mot réservé. C'est pourquoi il doit être répété devant chaque paramètre référence.

Un paramètre référence permet de retourner un résultat au module invoquant via l'argument correspondant. Par exemple, lorsqu'un module invoque **EXÉCUTER Interchanger(N1, N2)**, le module **Interchanger** (Pseudo-code 11-7) transpose en réalité le contenu des variables **N1** et **N2** données comme arguments. Consultez la section 11.5.3 pour obtenir plus d'information.

Notez que le module principal ne peut pas disposer de paramètres.

11.5.2 Paramètres valeurs

Lorsque les arguments dans une invocation sont des **variables**, le module invoqué travaille avec une copie de la variable servant d'argument, et ce même si les deux variables ont le même nom. Le contenu de la variable servant d'argument n'est pas changé. Le paramètre du module invoqué peut être modifié par ce dernier, mais cela n'affecte pas la valeur de la variable en argument.

L'exemple suivant illustre cette indépendance entre les arguments et les paramètres :

```

\\ Module auxiliaire SansChangement
ENTRER Lettre, Nombre
    ÉCRIRE Lettre, Nombre    \\ Affiche B et 12 (voir invocation)
    Lettre = 'A'
    Nombre = 32
    ÉCRIRE Lettre, Nombre    \\ Affiche A et 22
RETOURNER

```

```

\\ Module principal
DÉBUT
  Lettre = 'B'
  Nombre = 12
  ÉCRIRE Lettre, Nombre           \\ Affiche B et 12
  EXÉCUTER SansChangement(Lettre, Nombre)
  ÉCRIRE Lettre, Nombre           \\ Affiche B et 12
FIN

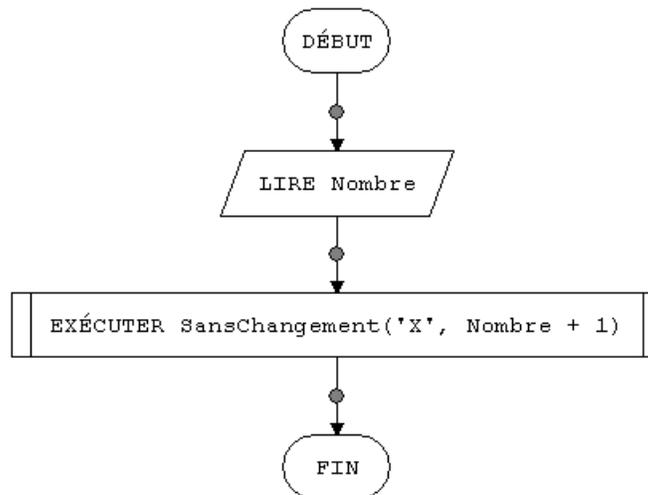
```

Pseudo-code 11-8 : Passage d'arguments par valeur

Dans le [module principal](#) ci-dessus, les variables **Lettre** et **Nombre** ne sont pas modifiées par l'invocation du [module auxiliaire SansChangement](#), et ce même si les paramètres ont des noms identiques aux arguments.

Lorsqu'un module reçoit des valeurs de variables fournies en arguments sans pouvoir modifier le contenu de ces variables, on appelle ces paramètres des *paramètres valeurs*. Dans la littérature technique on appelle ce type de transfert d'information la *transmission par valeur*. Dans LARP, les paramètres d'un module sont par défaut des paramètres valeurs.

Puisqu'un argument est indépendant d'un paramètre valeur correspondant, l'argument n'a pas à être exclusivement une variable; il peut être une constante ou le résultat d'une expression :



Organigramme 11-7 : Passage d'arguments par valeur

Dans cet exemple, le premier argument est une constante alors que le deuxième est le résultat d'une expression.

Contrairement aux paramètres valeurs qui peuvent recevoir comme argument des variables ou des expressions, les [paramètres références](#) doivent absolument recevoir des variables comme argument.

11.5.3 Paramètres références

Lorsqu'un paramètre énuméré dans l'instruction **ENTRER** d'un [module auxiliaire](#) est précédé du mot réservé **RÉFÉRENCE**, ce paramètre est un *paramètre référence*. Contrairement au [paramètre valeur](#) qui reçoit la valeur de l'argument correspondant lors de l'appel, le paramètre

référence fait référence à la même variable que celle passée comme argument lors de l'appel, et ce même si la variable argument est nommée différemment du paramètre référence.

Cette distinction entre les paramètres valeurs et les paramètres références est mieux expliquée via un exemple. Considérons les deux modules auxiliaires du Pseudo-code 11-9 : les deux modules ont exactement les mêmes instructions à l'exception de la définition des paramètres, où **Interchanger_1** exploite des paramètres valeurs alors que **Interchanger_2** exploite des paramètres références.

```

\\ Module auxiliaire Interchanger_1
ENTRER V1, V2
    Temp = V1
    V1   = V2
    V2   = Temp
RETOURNER

\\ Module auxiliaire Interchanger_2
ENTRER RÉFÉRENCE V1, RÉFÉRENCE V2
    Temp = V1
    V1   = V2
    V2   = Temp
RETOURNER

```

Pseudo-code 11-9 : Déclaration de paramètres valeurs et références

Le module principal suivant fait appel aux deux modules auxiliaires du Pseudo-code 11-9. Alors que l'appel au module **Interchanger_1** n'affecte en rien le contenu des arguments **N1** et **N2**, l'appel au module **Interchanger_2** change le contenu de ces deux variables :

```

\\ Module principal
DÉBUT
    N1 = 10
    N2 = 20

    EXÉCUTER Interchanger_1(N1, N2)
    ÉCRIRE N1, N2                \\ Affiche 10 20

    EXÉCUTER Interchanger_2(N1, N2)
    ÉCRIRE N1, N2                \\ Affiche 20 10
FIN

```

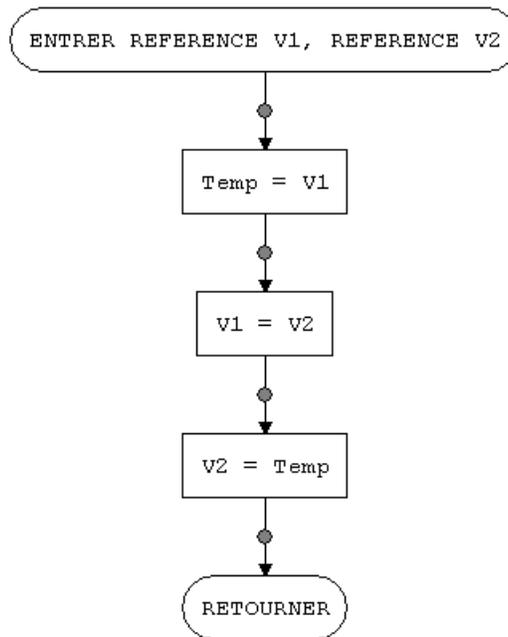
Pseudo-code 11-10 : Module principal

Puisque le module auxiliaire **Interchanger_2** exploite des paramètres références, les variables paramètres **V1** et **V2** sont en fait des synonymes des arguments **N1** et **N2**, respectivement. Ainsi, lorsque le module **Interchanger_2** affecte une valeur à **V1**, il affecte en réalité cette valeur à la variable **N1** du module principal. Il en est de même avec **V2** et **N2**. Cette relation entre les paramètres du module auxiliaire (**V1** et **V2**) et les arguments fournis en appel (**N1** et **N2**) n'existe pas lors de l'appel au module auxiliaire **Interchanger_1** puisque ce dernier exploite des paramètres valeurs : les valeurs des arguments (en occurrence les valeurs dans **N1** et **N2**) sont copiées dans les paramètres valeurs **V1** et **V2** ; lorsque le module **Interchanger_1** modifie le contenu de ses paramètres **V1** et **V2**, ceci n'affecte en rien le contenu des arguments **N1** et **N2**.

Cette relation entre les paramètres références et leurs arguments correspondants impose une restriction majeure sur ces mêmes arguments : *l'argument correspondant à un paramètre référence doit être une variable*. En effet, puisque le paramètre référence est une variable qui fait référence à l'argument (i.e. un synonyme), ce dernier doit aussi être une variable. Cette

restriction ne s'applique pas aux paramètres valeurs car ceux-ci reçoivent la valeur de leur argument, mais ne maintiennent pas une relation de référence avec celui-ci.

Les paramètres références sont identifiés à l'aide du mot réservé **RÉFÉRENCE** en en-tête du module auxiliaire, tel qu'illustré dans le module en pseudo-code **Interchanger_2** (Pseudo-code 11-9). Il en est de même pour les modules exprimés sous forme d'organigramme :



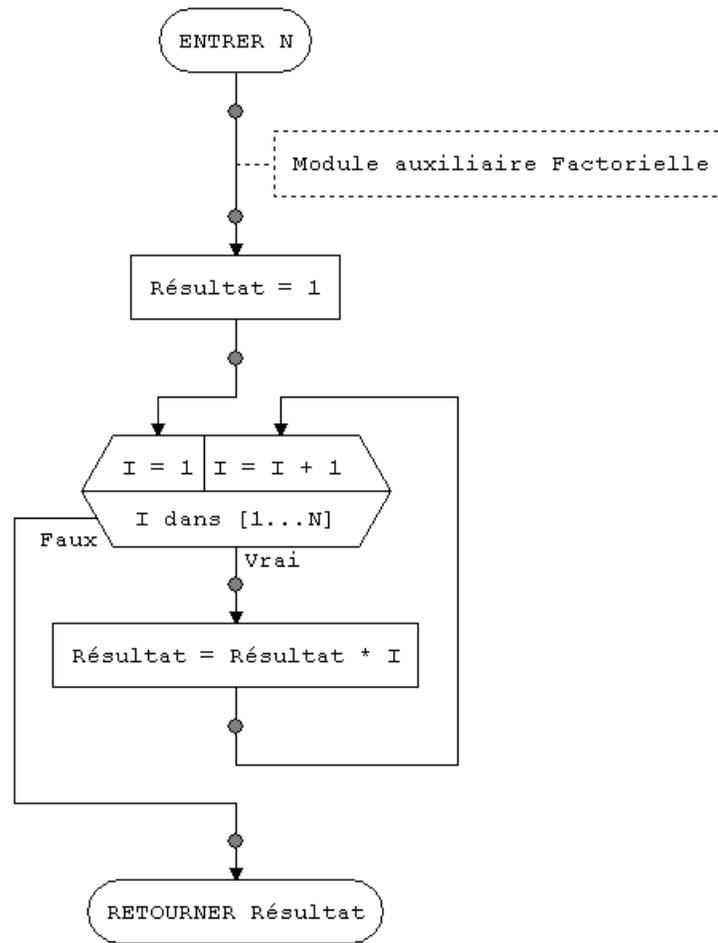
Organigramme 11-8 : Déclaration de paramètres références

Dans la littérature technique, la relation entre les paramètres références et les arguments fournis en appel est nommée la *transmission par référence*.

11.6 Modules avec valeur de retour

Les **modules auxiliaires** acceptent généralement des données en paramètres lors de l'exécution de l'algorithme. Les modules avec *valeur de retour* peuvent aussi avoir des paramètres, mais ont en plus la particularité de retourner un résultat au module invoquant via l'instruction **RETOURNER**. Les modules avec valeur de retour peuvent être utilisés pour effectuer des opérations nécessitant le retour d'un résultat, tels des opérations mathématiques complexes ou l'affichage d'un menu avec lecture de l'item sélectionnée par l'utilisateur.

Le module auxiliaire de l'Organigramme 11-9 calcule et retourne via valeur de retour la factorielle de son paramètre. La factorielle d'une valeur n est définie comme le résultat du produit de la séquence $1 \times 2 \times 3 \times \dots \times n$:



Organigramme 11-9 : Module auxiliaire avec valeur de retour

La valeur de retour du module est spécifiée à droite du mot réservé **RETOURNER**. La valeur de retour peut être le résultat d'une expression : **RETOURNER I+1**. Par contre une seule valeur de retour peut être spécifiée : **RETOURNER I,I+1** est erronée, mais **RETOURNER [I,I+1]** est acceptée car un [conteneur](#) est retourné.

Voici un autre exemple de module auxiliaire avec valeur de retour. Ce module affiche une menu, puis lit un numéro de commande jusqu'à ce qu'une commande valide soit sélectionnée, et enfin retourne le numéro de cette commande :

```

\\ Module auxiliaire Menu
ENTRER
RÉPÉTER
  ÉCRIRE "Le menu est"
  ÉCRIRE " 1 - Factorielle"
  ÉCRIRE " 2 - Addition"
  ÉCRIRE " 3 - Quitter"
  LIRE Commande
  JUSQU'À Commande >= 1 ET Commande <= 3
RETOURNER Commande
  
```

Pseudo-code 11-11 : Module de sélection d'une commande de menu

Lors de l'invocation d'un module comme ceux ci-dessus, la valeur de retour peut être récupérée dans une variable via l'[affectation](#) ou peut être directement exploitée dans une instruction *LARP*, tels que démontrés dans le module principal suivant :

```

\\ Module principal
DÉBUT
  Opération = EXÉCUTER Menu
  SI Opération = 1 ALORS
    LIRE N
    ÉCRIRE EXÉCUTER Factorielle(N)
  SINON SI Opération = 2 ALORS
    LIRE N1, N2
    ÉCRIRE N1+N2
  FINSI
FIN

```

Pseudo-code 11-12 : Invocation d'un module avec valeur de retour

Notez qu'un module auxiliaire peut retourner un seul résultat via la valeur de retour. Des [paramètres références](#) peuvent être exploités pour retourner plus d'un résultat.

11.7 Syntaxe d'invocation alternative

Un [module auxiliaire](#) est généralement invoqué à l'aide de l'instruction **EXÉCUTER**. *LARP* permet aussi d'invoquer un module sans le mot réservé **EXÉCUTER**, comme le démontre le module ci-dessous :

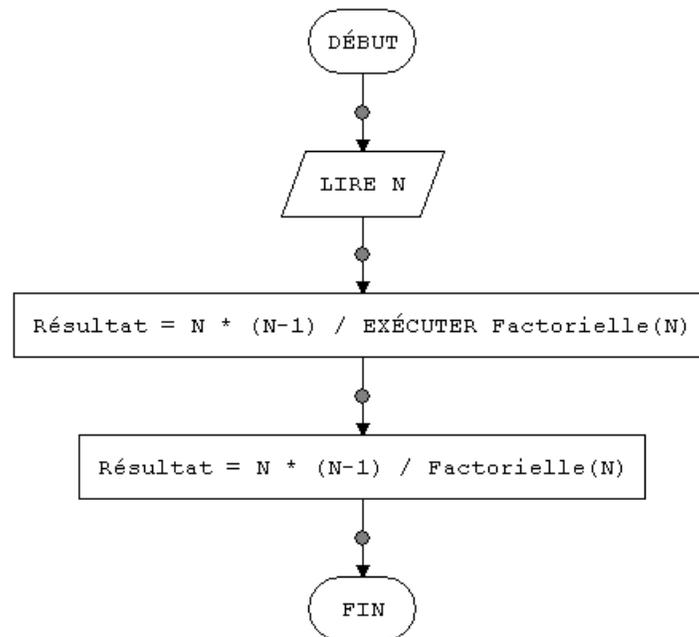
```

\\ Module principal
DÉBUT
  Opération = Menu // Invoque le module Menu
  SI Opération = 1 ALORS
    LIRE N
    ÉCRIRE Factorielle(N) // Invoque le module Factorielle
  SINON SI Opération = 2 ALORS
    LIRE N1, N2
    ÉCRIRE N1+N2
  FINSI
FIN

```

Pseudo-code 11-13 : Syntaxe alternative d'invocation d'un module

Par souci de lisibilité il est recommandé d'employer le mot réservé **EXÉCUTER** pour invoquer un module sans valeur de retour, mais de l'éviter lorsque le module invoqué retourne un résultat. Ainsi, dans l'exemple qui suit (Organigramme 11-10), la deuxième invocation du module auxiliaire **Factorielle** résulte en une instruction plus « élégante » que la première :



Organigramme 11-10 : Deux exemples d'invocation d'un même module

De plus, la seconde invocation est conforme à l'invocation de [fonctions prédéfinies](#) de *LARP* telle que l'invocation de **RACINE** :

```

\\ Module principal
DÉBUT
  LIRE N
  Résultat = RACINE(N) / Factorielle(N)
FIN
  
```

Pseudo-code 11-14 : Invocation d'une fonction prédéfinie

12 Fichiers et tampons d'entrées/sorties

Par défaut, lors de l'exécution d'un algorithme les données sont lues via le clavier et les résultats sont affichés à la [console d'exécution](#) (i.e. l'écran). Le clavier (en lecture) et la console d'exécution (en écriture) sont les *interfaces d'entrées/sorties standards* de LARP.

Dans certains cas cependant, l'algorithme doit exploiter des données provenant d'autres sources que le clavier. Ces données sont généralement stockées dans un document externe à l'algorithme. Dans d'autres cas, l'algorithme doit conserver les résultats produits dans un document externe permanent, ce qui n'est pas le cas avec la console d'exécution où les résultats affichés sont perdus lorsque la console est fermée.

LARP supporte deux sources d'informations externes :

- [Tampons d'entrées/sorties](#) : support de données intégré à un projet LARP. Ce support permet de lire et écrire directement dans un document intégré au projet LARP. Comme un module, un tampon d'entrées/sorties est identifié dans le projet par un nom unique.
- [Fichiers](#) : support de données généralement stocké sur le disque rigide de l'ordinateur (ou sur un autre ordinateur accessible via une connexion réseau). Le fichier est identifié par un nom unique au système de fichiers de l'ordinateur exécutant LARP.

Un algorithme qui lit des données ou écrit des résultats dans un fichier ou un tampon d'entrées/sorties accomplit une *opération d'entrée/sortie*. Un tel transfert d'information est effectué via un [canal d'entrées/sorties](#). Dans LARP, l'information traitée via un canal d'entrées/sorties est présentée sous forme textuelle (c'est-à-dire une séquence de caractères) dans le document visé. On considère un canal d'entrées/sorties comme une séquence de caractères. Les références à un canal d'entrées/sorties se font via un *numéro de canal*.

12.1 Tampons d'entrées/sorties

Un projet LARP est constitué d'au moins un [module principal](#) et peut-être un ou plusieurs [modules auxiliaires](#). LARP permet aussi de définir des « modules de données », communément appelés *tampons d'entrées/sorties*. Dans l'[environnement de développement](#) de LARP, les tampons d'entrées/sorties sont accessibles via le [navigateur de documents](#) (voir Figure 12-1).

Les tampons d'entrées/sorties sont créés par l'utilisateur de façon similaire à la création de modules auxiliaires :

- via les menus **Fichier** » **Nouveau...** et **Projet** » **Nouveau...** sur la [barre de menu](#), ou
- via le menu contextuel du navigateur de documents, accessible en cliquant le bouton droit de la souris lorsqu'elle est positionnée sur le navigateur.

Les règles régissant comment nommer un tampon sont identiques à celle dictant les [noms de modules](#).

Après avoir créé un tampon d'entrées/sorties, l'utilisateur peut y insérer des données via l'[éditeur textuel](#). L'algorithme peut ensuite lire ou écrire dans un tampon d'entrées/sorties du projet à l'aide des instructions [LIRE](#) et [ÉCRIRE](#). Notez que l'instruction [REQUÊTE](#) ne permet pas d'accéder au contenu des tampons d'entrées/sorties ni à celui des fichiers.

Note importante : la gestion des tampons d'entrées/sorties ne peut être faite que via l'environnement de développement de LARP. Il est impossible pour un algorithme de créer un nouveau tampon ou détruire un tampon existant durant son exécution.

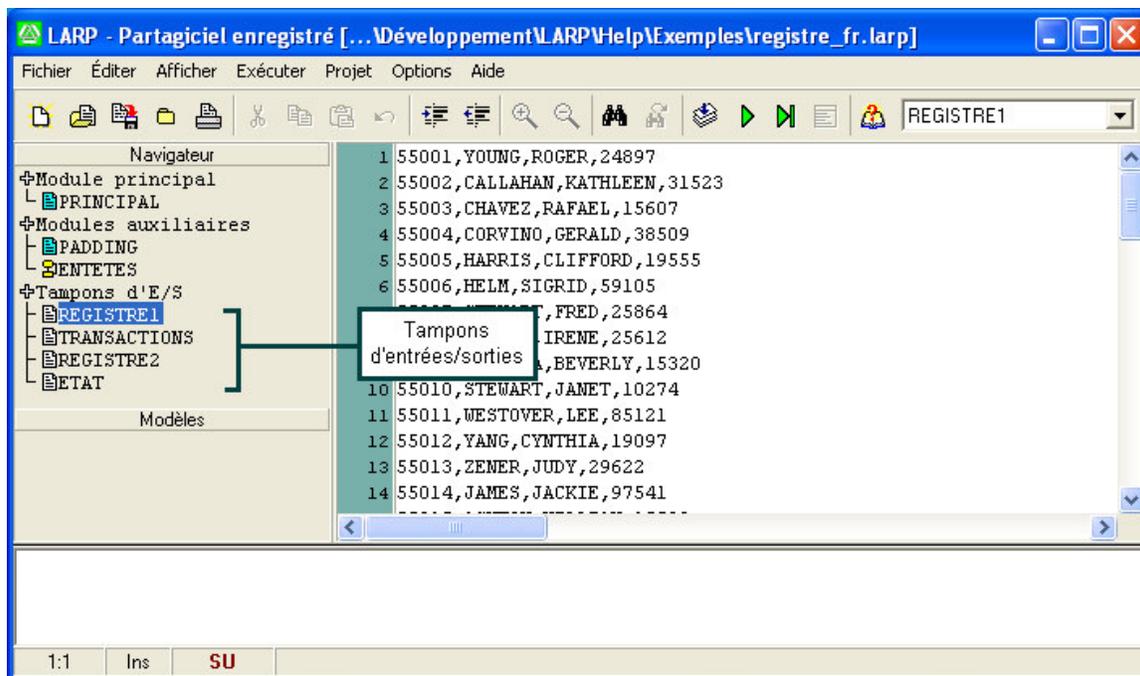


Figure 12-1 : Tampons d'entrées/sorties d'un projet

12.2 Fichiers

Comme pour les [tampons d'entrées/sorties](#), *LARP* peut accéder à des fichiers gérés par le système d'exploitation de l'ordinateur pour la lecture de données ou l'écriture de résultats. Voici les distinctions entre les tampons d'entrées/sorties et les fichiers :

- Les tampons d'entrées/sorties sont gérés par *LARP*, alors que les fichiers sont gérés par le système d'exploitation de l'ordinateur supportant *LARP*. C'est pourquoi les fichiers n'apparaissent pas dans le [navigateur de documents](#) de *LARP*, alors que les tampons d'entrées/sorties y figurent.
- Dans un algorithme, un tampon d'entrées/sorties est identifié par son nom lors de son ouverture. En plus de son nom, un fichier dispose d'un *chemin d'accès* indiquant la position du fichier (i.e. son *répertoire*) dans le système de fichiers de l'ordinateur. Ainsi, pour ouvrir un fichier, il faut fournir son nom ainsi que le chemin d'accès menant au répertoire où est situé le fichier.
- Un tampon d'entrées/sorties exploité dans un algorithme doit au préalable être créé via l'[environnement de développement](#) de *LARP*, alors qu'un fichier peut être implicitement créé par l'algorithme lors de l'exécution de celui-ci.
- Un tampon d'entrées/sorties n'est pas accessible à l'extérieur de l'environnement de développement de *LARP*. Par contre, un fichier est accessible en tout temps, par tout logiciel (*LARP* ou autres) exécutable sur l'ordinateur. Ainsi, un fichier créé à l'aide d'un autre logiciel peut être lu par un algorithme *LARP*, et un fichier créé par un algorithme *LARP* est accessible aux autres logiciels. Ce n'est pas le cas des tampons d'entrées/sorties. Notez cependant que l'environnement de développement de *LARP* permet de convertir un tampon d'entrées/sorties en fichier (via la commande **Projet** » **Tampons d'E/S** » **Exporter** de la [barre de menu](#)), et vice-versa (via la commande **Projet** » **Tampons d'E/S** » **Importer** de la barre de menu).

Du point de vue de l'algorithme exécuté, la seule distinction majeure entre un fichier et un tampon d'entrées/sorties est la façon d'ouvrir le document avec l'instruction **OUVRIR**.

12.3 Canaux d'entrées/sorties

L'exploitation d'un document (i.e. un **tampon d'entrées/sortie** ou un **fichier**) dans un algorithme *LARP* est effectuée via un *canal d'entrées/sorties*. *LARP* permet d'associer un numéro unique à chaque document lors de son ouverture. Par la suite toute instruction impliquant ce document exploite le numéro de canal d'entrées/sorties pour identifier le document visé.

LARP dispose de **256** canaux d'entrées/sorties, numérotés de 1 à 256. Ainsi, un algorithme *LARP* peut accéder simultanément à 256 tampons d'entrées/sorties et/ou fichiers. Lorsqu'un canal d'entrées/sorties est associé à un document (lors de son **ouverture**), cette association doit être explicitement rompue (par la **fermeture** du document) avant d'associer le canal à un autre document. Similairement, deux canaux d'entrées/sorties ne peuvent pas simultanément être associés à un même document. Toute violation à ces restrictions entraîne automatiquement une interruption de l'exécution de l'algorithme.

L'utilisation des canaux d'entrées/sorties est bien illustrée dans les exemples des prochaines sections.

12.3.1 Ouverture d'un document

L'instruction d'ouverture d'un document est **OUVRIR**. Cette instruction permet :

- d'associer un **canal d'entrées/sorties** à un fichier ou un tampon d'entrées/sorties, et
- d'indiquer le **mode d'accès** au canal d'entrées/sorties.

Un **tampon d'entrées/sorties** ou un **fichier** ne peut être associé à plus d'un canal simultanément, et il demeure associé au canal jusqu'à ce que celui-ci soit **fermé**.

À l'exception de l'instruction d'association de canal (**OUVRIR**), *LARP* ne fait aucune distinction entre les accès à un tampon d'entrées/sorties et à un fichier. Les instructions de gestion de canal (**LECTURE**, **ÉCRITURE**, **FERMER**, ...) de *LARP* utilisent le canal d'entrées/sorties comme référence au document ouvert.

12.3.2 Ouvrir un tampon d'entrées/sorties

Par défaut, l'instruction **OUVRIR** associe un **tampon d'entrées/sorties** au **canal d'entrées/sorties** spécifié :

```
\\ Ouvrir un tampon en mode Lecture
OUVRIR "DONNEES" SUR 3 EN LECTURE
```

Pseudo-code 12-1 : Ouvrir un tampon d'entrées/sorties

Cette instruction associe le canal 3 au tampon d'entrées/sorties nommé **DONNEES** afin d'en lire le contenu.

Par défaut, *LARP* associe une requête d'ouverture à un tampon. On peut explicitement spécifier le document comme étant un tampon. L'organigramme suivant est équivalent au pseudo-code précédent :

↓

```
OUVRIR TAMPON "DONNEES" SUR 3 EN LECTURE
```

|

Organigramme 12-1 : Exploitation du mot réservé TAMPON

Notez que l'instruction séquentielle d'organigramme est utilisée pour insérer une instruction d'ouverture de tampon d'entrées/sorties ou de fichier dans un organigramme.

Comme stipulé précédemment, il est impossible d'associer plus d'un canal à un même tampon. Il est de même impossible d'associer deux tampons d'entrées/sorties à un même canal :

```
\\ Ouvrir un tampon en mode Lecture
OUVRIR "DONNEES" SUR 3 EN LECTURE
OUVRIR "DONNEES" SUR 4 EN LECTURE    \\ Erreur: fichier déjà associé
                                         \\          au canal 3
OUVRIR "DATA" SUR 3 EN LECTURE      \\ Erreur: canal 3 occupé
```

Pseudo-code 12-2 : Instructions OUVRIR invalides

Il est important de noter qu'*un algorithme ne peut pas créer des tampons d'entrées/sorties lors de son exécution*. Un projet *LARP* doit donc au préalable disposer des tampons entrées/sorties requis pour l'exécution de ses modules. L'utilisateur doit créer les tampons via l'[environnement de développement](#) avant d'exécuter l'algorithme exploitant ceux-ci.

Les tampons d'entrées/sorties définis dans un projet *LARP* et leur contenu sont sauvegardés avec les modules dans le fichier *LARP* lors de la sauvegarde du projet.

12.3.3 Ouvrir un fichier

L'instruction permettant d'accéder à un [fichier](#) est semblable à celle ouvrant un [tampon d'entrées/sorties](#), avec le mot réservé **FICHER** remplaçant **TAMPON** :

```
\\ Ouvrir un fichier en mode Lecture
OUVRIR FICHER "C:\\DONNEES.TXT" SUR 3 EN LECTURE
```

Pseudo-code 12-3 : Ouvrir un fichier

Le pseudo-code ci-dessus ouvre en mode lecture le fichier **DONNEES.TXT** localisé dans le répertoire **C:**. Le chemin au fichier visé dans la structure de répertoires doit être spécifié à l'aide d'*antéslashs* (\), où le double antéslash est la [séquence d'échappement](#) représentant l'antéslash simple (\).

Si aucun chemin de répertoire n'est spécifié avec le nom du fichier, *LARP* ouvre le fichier dans le répertoire courant (habituellement le répertoire où est sauvegardé le projet *LARP* en exécution). Cependant, puisqu'il peut y avoir des exceptions à cette règle, il est recommandé de toujours précéder le nom d'un fichier du chemin de répertoire complet où est situé le fichier dans l'unité de stockage.

LARP peut être dans l'impossibilité d'ouvrir le fichier spécifié pour une ou l'autre des raisons suivantes :

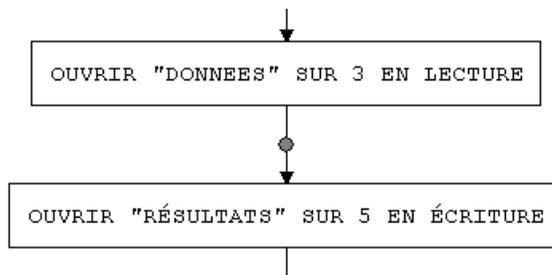
- **Le fichier n'existe pas** : l'algorithme tente d'ouvrir en mode lecture un fichier inexistant.

- **Le fichier est déjà ouvert** : l'algorithme tente d'ouvrir un fichier qui est déjà ouvert par LARP ou par une autre application de l'ordinateur.
- **Le canal d'entrées/sorties n'est pas disponible** : l'algorithme tente d'ouvrir un fichier sur un [canal d'entrées/sorties](#) invalide ou déjà associé à un autre document.
- **Le nom du fichier est invalide** : le nom de fichier spécifié est invalide (peut être dû au répertoire inexistant, au nom du fichier contenant des caractères interdits par *Windows*[®] ou à l'unité de stockage défectueuse ou non disponible).

Lorsqu'il y a erreur à l'ouverture d'un fichier, LARP interrompt l'exécution de l'algorithme et affiche un [message d'erreur](#) précisant la cause de l'erreur.

12.3.4 Modes d'accès

Lors de l'ouverture d'un [tampon d'entrées/sorties](#) ou d'un [fichier](#), un mode d'accès doit être spécifié :



Organigramme 12-2 : Spécification du mode d'accès à un document

Trois modes d'accès aux tampons d'entrées/sorties et fichiers sont supportés par LARP :

- **LECTURE** : permet de [lire](#) le contenu du document à l'aide de l'instruction **LIRE**. Si le document est inexistant, l'exécution de l'algorithme est interrompue.
- **ÉCRITURE** : permet d'[écrire](#) des résultats dans le document à l'aide de l'instruction **ÉCRIRE**. Le contenu antérieur à l'ouverture du document est effacé. Si le document est un fichier inexistant, celui-ci est créé. Si le document est un tampon d'entrées/sorties inexistant, l'exécution de l'algorithme est interrompue.
- **AJOUT** : permet d'écrire des résultats à *la fin* du document à l'aide de l'instruction **ÉCRIRE**. Le contenu antérieur à l'ouverture du document est conservé. Si le document est un fichier inexistant, celui-ci est créé. Si le document est un tampon d'entrées/sorties inexistant, l'exécution de l'algorithme est interrompue.

La principale distinction entre les modes d'accès **ÉCRITURE** et **AJOUT** est dans la gestion du contenu antérieur du document :

- L'ouverture d'un tampon d'entrées/sorties ou d'un fichier en mode **ÉCRITURE** efface automatiquement le contenu antérieur du document.
- L'ouverture d'un tampon d'entrées/sorties ou d'un fichier en mode **AJOUT** préserve le contenu existant, les nouveaux résultats y étant écrits à la fin.

Seule l'instruction de lecture (**LIRE**) est autorisée sur un [canal d'entrées/sorties](#) associé à un document ouvert en mode **LECTURE**. Similairement, seule l'instruction d'écriture (**ÉCRIRE**) est autorisée sur un canal d'entrées/sorties associé à un document ouvert en mode **ÉCRITURE** ou

AJOUT. Toute instruction de lecture ou d'écriture invalide sur un canal d'entrées/sorties cause automatiquement l'arrêt de l'exécution de l'algorithme et un [message d'erreur](#) approprié est affiché.

12.3.5 Fermeture d'un canal d'entrées/sorties

Tout [tampon d'entrées/sorties](#) ou [fichier](#) ouvert doit éventuellement être fermé. L'instruction de fermeture est appliquée au [canal d'entrées/sorties](#) associé au document qu'on désire fermer :

```
OUVRIR "DONNEES" SUR 3 EN LECTURE
FERMER 3
```

Pseudo-code 12-4 : Fermer un tampon d'entrées/sorties ou un fichier

Voici les règles régissant la fermeture des documents (tampons d'entrées/sorties et fichiers) :

- Tout document ouvert doit être fermé. Si l'algorithme termine son exécution sans avoir fermé tous les tampons d'entrées/sorties et fichiers ouverts, un [message d'avertissement](#) est affiché dans le [panneau de messages](#) et les documents ouverts sont automatiquement fermés par *LARP*.
- L'instruction de fermeture (**FERMER**) ne fait aucune distinction entre les [modes d'accès](#) (i.e. les canaux ouverts en mode **LECTURE**, **ÉCRITURE** et **AJOUT** sont fermés de façon identique).
- Un document ouvert doit être fermé qu'une seule fois (une deuxième instruction de fermeture visant un même canal cause l'arrêt d'exécution de l'algorithme).
- Une instruction **FERMER** impliquant un canal d'entrées/sorties invalide (par exemple, un canal non associé à un tampon d'entrées/sorties ou un fichier) cause l'arrêt d'exécution de l'algorithme.

L'instruction **FERMER** permet de fermer plus d'un canal d'entrées/sorties. Ceux-ci doivent être énumérés en les séparant par des virgules :

```
FERMER 3, 5, 6
```

Organigramme 12-3 : Fermer plusieurs canaux d'entrées/sorties

12.4 Lecture via un canal d'entrées/sorties

La lecture de données à partir d'un [tampon d'entrées/sorties](#) ou d'un [fichier](#) se fait via le [canal d'entrées/sorties](#) associé au document visé :

```
OUVRIR "DONNEES" SUR 3 EN LECTURE
LIRE Nom, Numéro, Salaire DE 3
```

Pseudo-code 12-5 : Lecture d'un document via un canal d'entrées/sorties

La syntaxe d'une instruction **LIRE** visant un canal d'entrées/sorties est semblable à une instruction de [lecture visant le clavier](#); il faut cependant spécifier le canal d'entrées/sorties visé à l'aide du mot réservé **DE** dans un pseudo-code ou de spécifier le numéro de canal dans la *fenêtre d'édition d'instruction d'organigramme* :

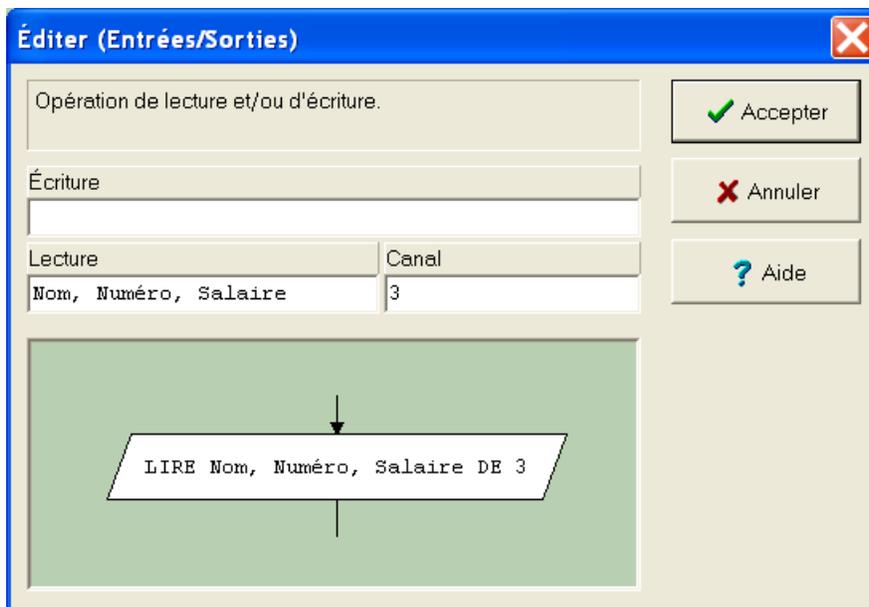


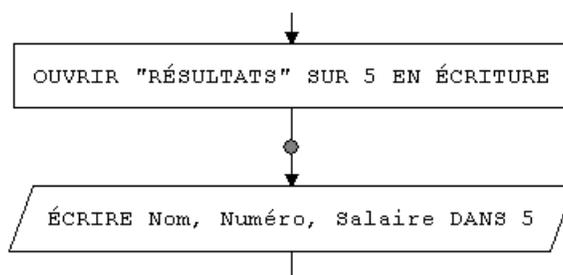
Figure 12-2 : Spécifier un numéro de canal dans une instruction de lecture d'organigramme

Le canal d'entrées/sorties doit obligatoirement être associé à un document (tampon d'entrées/sorties ou fichier) ouvert en **mode LECTURE**. Toute tentative de lecture visant un canal d'entrées/sorties associé à un document ouvert en mode **ÉCRITURE** ou **AJOUT** cause l'arrêt d'exécution de l'algorithme.

Comme pour les instructions de lecture visant le clavier, les instructions de lecture visant les canaux d'entrées/sorties sont sujettes au **séparateur** courant.

12.5 Écriture via un canal d'entrées/sorties

L'écriture de résultats dans un **tampon d'entrées/sorties** ou dans un **fichier** se fait via le **canal d'entrées/sorties** associé au document visé :



Organigramme 12-4 : Écriture dans un document via un canal d'entrées/sorties

La syntaxe d'une instruction **ÉCRIRE** visant un canal d'entrées/sorties est semblable à une instruction d'écriture visant la **console d'exécution**; il faut cependant spécifier le canal d'entrées/sorties visé à l'aide du mot réservé **DANS** dans un pseudo-code ou de spécifier le numéro de canal dans la **fenêtre d'édition d'instruction d'organigramme** :

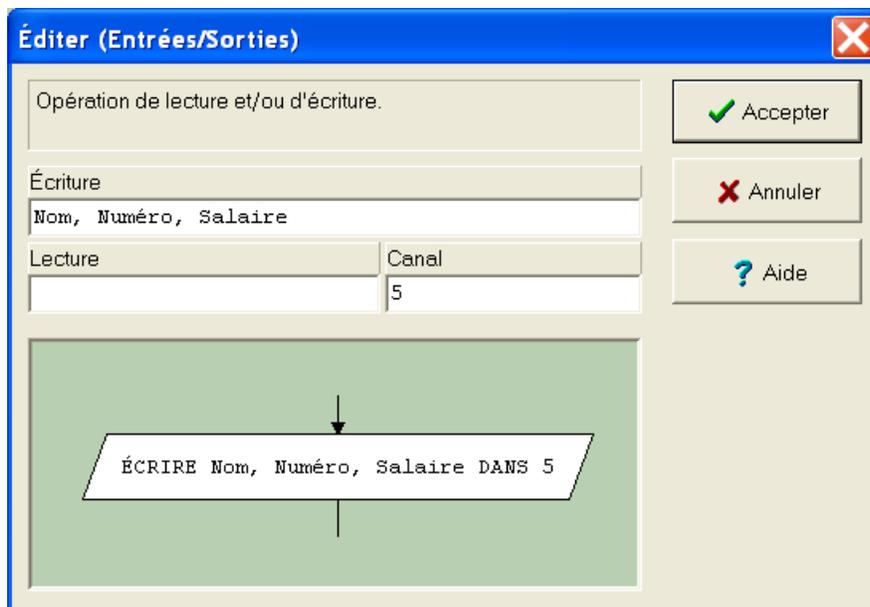


Figure 12-3 : Spécifier un numéro de canal dans une instruction d'écriture d'organigramme

Le canal d'entrées/sorties doit obligatoirement être associé à un document (tampon d'entrées/sorties ou fichier) ouvert en **mode ÉCRITURE** ou **AJOUT**. Toute tentative d'écriture visant un canal d'entrées/sorties associé à un document ouvert en **mode LECTURE** cause l'arrêt d'exécution de l'algorithme.

Comme pour les instructions d'écriture visant la console d'exécution, les instructions d'écriture visant les canaux d'entrées/sorties sont sujettes au **séparateur** courant.

12.6 Détection de fin de contenu via un canal d'entrées/sorties

Puisqu'un **tampon d'entrées/sorties** ou un **fichier** contient une quantité finie de données, il est possible de déterminer si la dernière opération de lecture a atteint la fin des données. La fonction prédéfinie **FINDECONTENU** appliquée à un **canal d'entrées/sorties** permet de déterminer dans une condition de **structure conditionnelle** ou **répétitive** si la fin du document lu est atteinte :

```
Somme = 0
OUVRIR "DONNEES" SUR 3 EN LECTURE
RÉPÉTER
  LIRE Valeur DE 3
  Somme = Somme + Valeur
JUSQU'À FINDECONTENU(3)
FERMER 3
ÉCRIRE Somme
```

Pseudo-code 12-6 : Détection de la fin d'un document lors d'une lecture

La fonction prédéfinie **FINDECONTENU** n'est applicable qu'aux canaux d'entrées/sorties associés en **mode lecture**. Toute invocation de la fonction **FINDECONTENU** sur un canal d'entrées/sorties associé à un document ouvert en **mode ÉCRITURE** ou **AJOUT** cause l'arrêt d'exécution de l'algorithme.

La fonction **FINDECONTENU** n'est pas applicable au clavier puisqu'il est impossible d'associer un canal d'entrées/sorties à celui-ci.

Annexe A - Calcul binaire

Les ordinateurs calculent d'une façon différente de l'humain. Alors que nous calculons généralement avec des nombre décimaux (base 10, avec les chiffres 0 à 9), les ordinateurs calculent avec des nombres *binaires* (base 2, avec les chiffres 0 et 1).

A.1 Pourquoi les ordinateurs sont-ils binaires?

Malgré le fait que les ordinateurs sont des machines capables de traiter du texte, de jouer de la musique ou de projeter des vidéos, en réalité ils ne sont capables que d'une seule chose : faire des calculs, et uniquement cela. Ce sont en réalité des calculatrices améliorées.

L'ordinateur représente et traite du texte, du son, des images et de la vidéo sous forme de nombres. On dit qu'il manipule des *informations binaires*. L'information binaire est une unité d'information ne pouvant avoir que deux états : *activé/désactivé*. Cette représentation a comme origine les supports physiques permettant de stocker l'information. Puisque ces supports fonctionnent à l'électricité, seules la présence ou l'absence de courant dans un *transistor* permettent de représenter une unité d'information.

On symbolise une information binaire, quel que soit son support physique, sous la forme de 1 et de 0. Ainsi, l'ordinateur utilise comme base arithmétique les chiffres 0 et 1 (contrairement aux humains qui utilisent les chiffres 0 à 9). Malgré cette limitation, l'ordinateur est tout de même capable d'effectuer des calculs aussi complexes que les humains. En fait, cette limitation en terme de base de calcul est amplement compensée par la vitesse de calcul de l'ordinateur. C'est pourquoi ce dernier semble si performant par rapport à l'humain, même si à la base il ne sait manipuler que des zéros et des uns.

A.2 La numérotation en base décimale

Pour représenter un nombre, nous utilisons la *numérotation de position en base décimale*. La base décimale met à notre disposition un alphabet de dix chiffres (0 à 9). Lorsque nous écrivons un nombre avec ces chiffres, l'ordre dans lequel nous mettons les chiffres est primordial. Ainsi, 3498 n'est pas du tout le même nombre que 8439. Cet ordonnancement est la deuxième caractéristique de notre système de notation numérique: sa base *décimale*.

Ainsi, le nombre 3498 peut être décomposé en fonction de la position de chaque chiffre dans le nombre :

3498, c'est $3000 + 400 + 90 + 8$

Les zéros apparaissant dans cette décomposition proviennent d'un facteur multiplicatif permettant de positionner chaque chiffre dans le nombre :

3000	c'est 3×1000
400	c'est 4×100
90	c'est 9×10
8	c'est 8×1

On peut donc écrire le nombre 3498 de multiples façons :

$$3498 = 3 \times 1000 + 4 \times 100 + 9 \times 10 + 8 \times 1$$

ou bien :

$$3498 = (3 \times 10 \times 10 \times 10) + (4 \times 10 \times 10) + (9 \times 10) + (8 \times 1)$$

ou encore :

$$3498 = 3 \times 10^3 + 4 \times 10^2 + 9 \times 10^1 + 8 \times 10^0$$

Voilà donc le mécanisme général de la représentation par numérotation de position en base décimale. Comme on le constate dans la dernière représentation du nombre 3498, les chiffres de ce nombre sont positionnés en fonction d'une puissance de 10, de droite vers la gauche, en commençant par la puissance 0.

A.3 La numérotation en base binaire

La technologie permettant de stocker et manipuler de l'information dans un ordinateur est rudimentaire; l'information y est stockée sous forme binaire, par paquets de 0 et de 1. Par convention, la taille de ces paquets est de 8 informations binaires. Une information binaire (symbolisée par 0 ou 1) s'appelle un *bit*. Un groupe de huit bits est un *octet* (*byte* en anglais).

Combien d'états un octet peut-il posséder ? Retournons momentanément à la base décimale. Combien de nombres peut-on représenter avec *trois* chiffres en base décimale ? En fait, trois chiffres en base 10 permettent de représenter 10^3 ou 1000 nombres (i.e. de 0 à 999). De façon analogue, un octet contenant 8 bits en base 2 peut représenter $2^8 = 256$ nombres. Un octet permet donc de coder les nombres 0 à 255 en binaire (ou de -127 à +128 si les nombres négatifs doivent être représentés). Pour représenter de plus grands nombres, plus d'un octet sont requis. Ainsi, deux octets (i.e. 16 bits) permettent de représenter $2^{16} = 65536$ nombres, trois octets permettent de représenter $2^{24} = 16777216$ nombres, et ainsi de suite.

Puisque l'ordinateur doit manipuler divers types d'information, un octet peut servir à coder autre chose qu'un nombre, comme par exemple du texte. Puisqu'il y a seulement 26 lettres dans l'alphabet, même en comptant les majuscules, les minuscules, les caractères accentués ainsi que les chiffres et les signes de ponctuation, le nombre total de caractères à représenter est tout de même inférieur à 256. Ainsi, un seul octet peut être employé pour représenter distinctivement n'importe quel caractère conventionnel d'un texte français.

Il existe un standard international et universel de codage des caractères, chiffres et signes de ponctuation basé sur le codage binaire : le codage *ASCII* (pour *American Standard Code for Information Interchange*). Le codage ASCII établit une correspondance entre les caractères standards de l'alphabet (ainsi que d'autres symboles couramment employés en typographie) et les nombres binaires 0 à 255. Le Tableau 12-1 présente les codes ASCII correspondant aux caractères affichables couramment employés dans les algorithmes.

Revenons au codage des nombres sur un octet. La conversion d'un nombre binaire en nombre décimal correspondant est obtenue en appliquant la décomposition présentée précédemment en base décimale, mais cette fois-ci appliquée en base binaire. Prenons un octet au hasard :

1 0 0 1 1 1 0 1

Ce nombre représente en base 10, de gauche vers la droite :

$$\begin{aligned} & 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = \\ & 1 \times 128 + 1 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 1 = \\ & 128 + 16 + 8 + 4 + 1 = \\ & 157 \end{aligned}$$

ASCII	Lettre	ASCII	Lettre	ASCII	Lettre	ASCII	Lettre
032	Espace	064	@	096	`	128	Ç
033	!	065	A	097	a	129	ü
034	"	066	B	098	b	130	é
035	#	067	C	099	c	131	â
036	\$	068	D	100	d	132	ä
037	%	069	E	101	e	133	à
038	&	070	F	102	f	134	â
039	'	071	G	103	g	135	ç
040	(072	H	104	h	136	ê
041)	073	I	105	i	137	ë
042	*	074	J	106	j	138	è
043	+	075	K	107	k	139	ï
044	,	076	L	108	l	140	î
045	-	077	M	109	m	141	ì
046	.	078	N	110	n	142	Ä
047	/	079	O	111	o	143	Å
048	0	080	P	112	p	144	É
049	1	081	Q	113	q	145	æ
050	2	082	R	114	r	146	Æ
051	3	083	S	115	s	147	ô
052	4	084	T	116	t	148	ö
053	5	085	U	117	u	149	ò
054	6	086	V	118	v	150	û
055	7	087	W	119	w	151	ù
056	8	088	X	120	x	152	ÿ
057	9	089	Y	121	y	153	Ö
058	:	090	Z	122	z	154	Ü
059	;	091	[123	{	155	ø
060	<	092	\	124		156	£
061	=	093]	125	}	157	Ø
062	>	094	^	126	~	158	×
063	?	095	_	127		159	f

Tableau 12-1 : Table des codes ASCII

Inversement, pour convertir un nombre décimal en base binaire, il faut rechercher dans ce nombre les puissances successives de 2 à partir de la plus grande des puissances inférieures au nombre. Par exemple, 157 sera convertit ainsi :

- 157 contient 1 x 128 (2^7), et il reste 29.
- 29 contient 0 x 64 (2^6), et il reste toujours 29.
- 29 contient 0 x 32 (2^5), et il reste toujours 29.
- 29 contient 1 x 16 (2^4), et il reste 13.
- 13 contient 1 x 8 (2^3), et il reste 5.
- 5 contient 1 x 4 (2^2), et il reste 1.
- 1 contient 0 x 2 (2^1), et il reste toujours 1.
- 1 contient 1 x 1 (2^0), et finalement il ne reste rien.

En reportant ces résultats dans l'ordre de calcul, on obtient 10011101.

A.4 La numérotation hexadécimale

Puisque l'humain n'est généralement pas familier avec la notation binaire, il lui est difficile de "lire" un nombre binaire, i.e. convertir mentalement un nombre de la base binaire à la base décimale (par exemple, convertir 10011101 en base 2 à 157 en base 10 exige un calcul difficilement réalisable mentalement). Pour faciliter la lecture binaire à l'humain, un troisième codage est exploité couramment en informatique, le *codage hexadécimal*, en **base 16**.

La base hexadécimale représente l'octet non pas comme 8 bits mais comme deux paquets de 4 bits (les quatre bits de gauche, et les quatre bits de droite). Quatre bits permettent de coder $2^4 = 16$ nombres différents. Alternativement, un seul chiffre hexadécimal est requis pour représenter 16 nombres (de même qu'en base 10, un chiffre permet de représenter 10 nombres, i.e. de 0 à 9).

En base hexadécimale, aux dix premiers chiffres de la base décimale (0 à 9) sont ajoutées les 6 premières lettres de l'alphabet (A à F). Par convention, A vaut 10, B vaut 11, C vaut 12, D vaut 13, E vaut 14 et F vaut 15. Ainsi, les chiffres utilisés pour représenter les nombres en base hexadécimale sont 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E et F.

La table qui suit résume les correspondances décimales et binaires aux chiffres hexadécimaux :

Hexadécimal	Décimal	Binaire
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Tableau 12-2 : Correspondance des bases de numérotation

Puisque chaque chiffre hexadécimal peut représenter 16 nombres ou 4 bits ($2^4 = 16$), on peut ainsi utiliser deux chiffres hexadécimaux pour représenter un octet. Par exemple, comme nous l'avons vu précédemment le nombre 157 (en base décimale) est représenté ainsi en binaire :

1 0 0 1 1 1 0 1

Cet octet peut être représenté en base hexadécimale en divisant ses bits en deux paquets de 4 bits, puis en représentant chacun de ces deux paquets par le chiffre hexadécimal correspondant :

1 0 0 1 1 1 0 1
 9 D

Ainsi, le nombre binaire 10011101 est représenté en hexadécimal par *9D*, c'est-à-dire 157 en base décimale.

Le codage hexadécimal est couramment utilisé pour représenter les octets individuellement, la valeur d'un octet pouvant ainsi être représentée par deux chiffres (*9D*) en base hexadécimale plutôt qu'en huit chiffres (10011101) en base binaire. De plus, lorsque la table de conversion hexadécimale à binaire (Tableau 12-2) est mémorisée, il est facile pour le programmeur de convertir mentalement des nombre hexadécimaux en nombres binaires, et vice-versa.

Annexe B - Récursivité

En algorithmique, la programmation de **modules** permet d'implanter une logique empruntée aux mathématiques : la *programmation récursive*. Celle-ci est présentée dans cette annexe en prenant comme exemple le calcul de la factorielle mathématique.

La factorielle d'un nombre n , communément écrite **$n!$** , est définie selon la formule suivante :

$$n! = 1 \times 2 \times 3 \times \dots \times n$$

Cette équation peut être généralisée sous forme *récursive* :

$$n! = (n-1)! \times n$$

On dit cette équation récursive car la factorielle d'un nombre est définie en fonction de ce nombre multiplié par la factorielle du nombre précédent. Donc *la factorielle est définie à l'aide de la factorielle*. On semble tourner en rond, mais ce n'est pas le cas puisqu'on peut au préalable calculer $(n-1)!$ afin de calculer $n!$. Puisque $(n-1)! = (n-2)! \times (n-1)$, il faut au préalable calculer $(n-2)!$, et ainsi de suite. Puisque par définition $0! = 1$, le cycle de calcul de la factorielle du nombre précédent s'arrête à $n = 0$ dont on connaît la réponse, 1.

Dans *LARP* comme dans la plupart des langages de programmation, on peut écrire un module **Factorielle** qui calcule la factorielle d'un paramètre donné. Cette fonction effectue la multiplication du nombre fourni en paramètre par la factorielle du nombre précédent. Et cette factorielle du nombre précédent va bien entendu être elle-même calculée par la fonction **Factorielle**. Autrement dit, on crée une fonction qui *fait appel à elle même*. C'est la **récursivité**.

Voici un pseudo-code *LARP* récursif calculant la factorielle de son paramètre n :

```

\\ Module Factorielle
ENTRER n
  SI n = 0 ALORS
    nFac = 1
  SINON
    nFac = n * Factorielle(n-1)
  FINSI
RETOURNER nFac

```

Pseudo-code B-1 : Calcul de la factorielle (module récursif)

On remarque premièrement dans ce module que le processus d'appels récursifs est précédé d'une **condition** s'assurant que les appels récursifs vont éventuellement s'arrêter (lorsque n atteindra 0). Sans une telle condition, la récursivité serait appliquée indéfiniment. Ceci est une caractéristique commune à tous les modules récursifs : *ils doivent disposer d'une condition qui cessera éventuellement les appels récursifs*.

On peut aussi remarquer que le processus récursif remplace en quelque sorte une **structure répétitive**, c'est-à-dire un processus itératif. En fait, le module ci-dessus peut facilement être écrit sous une forme itérative plutôt que récursive, tel que démontré dans le pseudo-code suivant.

Ceci est une autre caractéristique des modules récursifs : *il est toujours possible d'écrire un module itératif équivalent qui n'exploite pas la récursivité*.

```
\\ Module Factorielle
ENTRER n
  nFac = 1
  TANTQUE n > 0 FAIRE
    nFac = nFac * n
    n = n - 1
  FINTANTQUE
RETOURNER nFac
```

Pseudo-code B-2 : Calcul de la factorielle (module itératif)

Pour conclure voici trois observations importantes sur la récursivité :

- La programmation récursive est très intuitive pour résoudre certains problèmes; elle permet d'écrire des modules simples et comportant peu d'opérations.
- Par contre la récursivité est très dispendieuse en ressources machine. Chaque appel récursif dans un module requiert de l'espace mémoire de l'ordinateur (car la valeur de chaque variable locale du module doit être temporairement stockée en mémoire avant de passer à l'appel récursif).
- Tout problème résolu par récursivité peut aussi l'être de façon itérative. En fait la récursivité n'est pas essentielle en programmation. Cependant elle est parfois plus élégante!

Annexe C - Fonctions prédéfinies

Cette section offre une description détaillée de chaque fonction prédéfinie de LARP.

Voici les fonctions prédéfinies disponibles :

ABSOLU	DATE	LOG10	PI
ALÉATOIRE	ENCARACTÈRES	LOGE	PLAFOND
ARCTANGENTE	ENCHAÎNE	LONGUEUR	PLANCHER
ARRONDIR	EXP	MAJUSCULES	POSITION
CAPACITÉ	FINDECONTENU	MAXIMUM	RACINE
COMPTER	FORMATER	MINIMUM	SINUS
COSINUS	HEURE	MINUSCULES	SOUSENSEMBLE

Notez que :

- Dans les descriptions de fonctions qui suivent , les crochets ([et]) sont généralement employés pour indiquer des éléments de syntaxe pouvant être omis.
- Les éléments de syntaxe présentés *en italique* sont descriptifs et ne sont pas partie intégrante de la syntaxe de l'appel de la fonction.

ABSOLU

Nom :	ABSOLU
Synonymes :	ABS
Type de retour :	Numérique
Nombre d'arguments :	1
Description :	ABSOLU retourne la valeur absolue (i.e. positive) du nombre donné en paramètre.
Format(s) d'appel :	ABSOLU (<i>numérique</i>)

Cette fonction retourne la valeur absolue de la valeur numérique fournie en paramètre. Le type de la valeur de retour correspond au type du paramètre (ex : si le paramètre est flottant, la valeur de retour est flottante).

Si le paramètre fourni est une *chaîne de caractères*, il y a tentative de conversion de celle-ci en valeur numérique.

Exemples

```
ÉCRIRE ABSOLU (-3.2)
ÉCRIRE ABS (17+4)
ÉCRIRE ABSOLU ("-4.1")
```

Pseudo-code C-1 : Exemples d'invocation de ABSOLU

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
3.2
21
4.1
```

Figure C-1 : Résultats de l'invocation de ABSOLU

ALÉATOIRE

Nom :	ALÉATOIRE
Synonymes :	ALÉA
Type de retour :	Numérique
Nombre d'arguments :	0, 1 ou 2
Description :	ALÉATOIRE retourne un nombre flottant ou entier choisi au hasard (plusieurs versions de la fonction sont disponibles).
Format(s) d'appel :	ALÉATOIRE ALÉATOIRE (numérique) ALÉATOIRE (numérique, numérique)

Cette fonction retourne une valeur numérique choisie au hasard. La valeur retournée dépend des paramètres fournis :

- **Aucun paramètre** : **ALÉATOIRE** retourne une valeur flottante x telle que $0 \leq x < 1$.
- **Un paramètre** : **ALÉATOIRE**(p) retourne une valeur x telle que $0 \leq x < p$. Le type de x correspond au type de p .
- **Deux paramètres** : **ALÉATOIRE**(p, q) retourne une valeur x telle que $p \leq x \leq q$. Le type de x correspond aux types de p et q (si p ou q est flottant, la valeur de retour x est flottante; si p et q sont entiers, la valeur de retour est entière).

Si un des paramètres fournis est une [chaîne de caractères](#), il y a tentative de conversion de celle-ci en valeur numérique.

Exemples

```
POUR i = 1 JUSQU'À 5 FAIRE
  ÉCRIRE "ALÉA= ", ALÉATOIRE
  ÉCRIRE "ALÉA(4)= ", ALÉATOIRE(4)
  ÉCRIRE "ALÉA(1,5.0)=", ALÉA(1, 5.0)
FINPOUR
```

Pseudo-code C-2 : Exemples d'invocation de ALÉATOIRE

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
ALÉA= 0.232930421130732
ALÉA(4) = 3
ALÉA(1,5.0) = 3.8551177335903
ALÉA= 0.438837686553597
ALÉA(4) = 3
ALÉA(1,5.0) = 1.61782418005168
ALÉA= 0.0393810363020748
ALÉA(4) = 0
ALÉA(1,5.0) = 1.03878780175
ALÉA= 0.378834913019091
ALÉA(4) = 1
ALÉA(1,5.0) = 2.45533445477486
ALÉA= 0.0822982566896826
ALÉA(4) = 1
ALÉA(1,5.0) = 3.0188071122393
```

Figure C-2 : Résultats de l'invocation de ALÉATOIRE

ARCTANGENTE

Nom :	ARCTANGENTE
Synonymes :	ARCTAN
Type de retour :	Numérique flottant
Nombre d'arguments :	1
Description :	ARCTANGENTE retourne la fonction trigonométrique inverse Tan^{-1} de l'angle donnée en paramètre (en radians).
Format(s) d'appel :	ARCTANGENTE(<i>numérique</i>)

Cette fonction retourne l'arc tangente (Tan^{-1}) de la valeur numérique (angle en radians) fournie en paramètre. La valeur de retour est de type flottant.

Si le paramètre fourni est une [chaîne de caractères](#), il y a tentative de conversion de celle-ci en valeur numérique.

Notez que diverses fonctions trigonométriques peuvent être calculées à l'aide des fonctions prédéfinies [SINUS](#), [COSINUS](#) et [ARCTANGENTE](#) :

$$Tan(x) = SINUS(x) / COSINUS(x)$$

$$Sin^{-1}(x) = ARCTANGENTE(x / RACINE(1 - (x * x)))$$

$$Cos^{-1}(x) = ARCTANGENTE(RACINE(1 - (x * x)) / x)$$

Exemples

```
ÉCRIRE ARCTANGENTE(1.2)
ÉCRIRE ARCTAN("-1.7")
```

Pseudo-code C-3 : Exemples d'invocation de ARCTANGENTE

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
0.876058050598193
-1.03907225953609
```

Figure C-3 : Résultats de l'invocation de ARCTANGENTE

ARRONDIR

Nom :	ARRONDIR
Synonymes :	ARR
Type de retour :	Numérique entier
Nombre d'arguments :	1
Description :	ARRONDIR retourne son paramètre arrondi à l'entier le plus près.
Format(s) d'appel :	ARRONDIR(<i>numérique</i>)

Cette fonction retourne son paramètre arrondi à l'entier le plus près. Si le paramètre est déjà entier, il est simplement retourné. Si par contre le paramètre est flottant, il est converti en entier le plus près; et si le flottant est exactement à mi-chemin entre deux entiers (i.e. $x + 0.5$), il est arrondi à l'entier le plus grand en magnitude absolue.

Si le paramètre fourni est une [chaîne de caractères](#), il y a tentative de conversion de celle-ci en valeur numérique.

Exemples

```
ÉCRIRE ARRONDIR(11.32)
ÉCRIRE ARR("-1.5")
```

Pseudo-code C-4 : Exemples d'invocation de ARRONDIR

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
11
-2
```

Figure C-4 : Résultats de l'invocation de ARRONDIR

CAPACITÉ

Nom :	CAPACITÉ
Synonymes :	CAP
Type de retour :	Numérique entier
Nombre d'arguments :	1
Description :	CAPACITÉ retourne le nombre d'éléments définis et indéfinis retrouvés dans le conteneur .
Format(s) d'appel :	CAPACITÉ (<i>conteneur</i>)

Cette fonction compte le nombre total de positions (i.e. les éléments définis et indéfinis) dans le conteneur fourni en paramètre.

Cette fonction n'est pas appliquée récursivement aux dimensions d'un conteneur multidimensionnel, le compte des éléments se faisant uniquement à la première dimension du conteneur.

Contrairement à la fonction **CAPACITÉ**, la fonction **COMPTER** retourne uniquement le nombre d'éléments définis dans un conteneur.

Exemples

```
ÉCRIRE CAPACITÉ ([1, , 3, ])
a = [1, [2, 3], 4]
ÉCRIRE CAP (a)
```

Pseudo-code C-5 : Exemples d'invocation de CAPACITÉ

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
4
3
```

Figure C-5 : Résultats de l'invocation de CAPACITÉ

COMPTER

Nom :	COMPTER
Synonymes :	COMPT, LONGUEUR
Type de retour :	Numérique entier
Nombre d'arguments :	1
Description :	COMPTER retourne le nombre d'éléments définis dans le conteneur , ou le nombre de caractères dans une chaîne de caractères .
Format(s) d'appel :	COMPTER (<i>conteneur</i>) COMPTER (<i>chaîne de caractères</i>)

Si le paramètre est une chaîne de caractères, cette fonction compte le nombre de caractères dans la chaîne.

Si le paramètre est un conteneur, cette fonction compte le nombre total d'éléments définis dans le conteneur. Cette fonction n'est pas appliquée récursivement aux dimensions d'un conteneur multidimensionnel, le compte des éléments se faisant uniquement dans la première dimension du conteneur.

Contrairement à la fonction **COMPTE**, la fonction **CAPACITÉ** retourne le nombre d'éléments définis *et indéfinis* dans un conteneur.

Exemples

```
ÉCRIRE COMPTE([1, , 3, ])
a = [1, [2, 3], 4]
ÉCRIRE COMPTE(a)
ÉCRIRE LONGUEUR("Bonjour le monde!")
```

Pseudo-code C-6 : Exemples d'invocation de COMPTE

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
2
3
17
```

Figure C-6 : Résultats de l'invocation de COMPTE

COSINUS

Nom :	COSINUS
Synonymes :	COS
Type de retour :	Numérique flottant
Nombre d'arguments :	1
Description :	COSINUS retourne la valeur de la fonction trigonométrique Cos appliquée à l'angle donnée en paramètre (en radians).
Format(s) d'appel :	COSINUS (numérique)

Cette fonction retourne le cosinus (*Cos*) de la valeur numérique (angle en radians) fournie en paramètre. La valeur de retour est de type flottant.

Si le paramètre fourni est une [chaîne de caractères](#), il y a tentative de conversion de celle-ci en valeur numérique.

Exemples

```
ÉCRIRE COSINUS(1 + 0.2)
ÉCRIRE COS("-1.7")
```

Pseudo-code C-7 : Exemples d'invocation de COSINUS

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
0.362357754476674
-0.128844494295525
```

Figure C-7 : Résultats de l'invocation de COSINUS

DATE

Nom :	DATE
Synonymes :	Aucun
Type de retour :	Conteneur
Nombre d'arguments :	0
Description :	DATE retourne un conteneur de quatre valeurs : l'année courante, le mois courant (1 à 12), le jour courant (1 à 31) et le jour de la semaine (1 à 7).
Format(s) d'appel :	DATE

Cette fonction retourne la date courante dans un conteneur. Les quatre valeurs contenues dans le conteneur retourné sont :

DATE[1] est l'année courante,

DATE[2] est le mois courant (1 = janvier à 12 = décembre),

DATE[3] est le jour courant, et

DATE[4] est le jour de la semaine (1 = dimanche jusqu'à 7 = samedi).

Exemples

```
ÉCRIRE DATE
ÉCRIRE "Année =", DATE[1]
```

Pseudo-code C-8 : Exemples d'invocation de DATE

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
[2006 3 13 2]
2006
```

Figure C-8 : Résultats de l'invocation de DATE

ENCARACTÈRES

Nom :	ENCARACTÈRES
Synonymes :	ENCARAC
Type de retour :	Conteneur
Nombre d'arguments :	1
Description :	ENCARACTÈRES convertit le paramètre donné en un conteneur ayant chaque caractère comme élément distinct.
Format(s) d'appel :	ENCARACTÈRES (chaîne de caractères) ENCARACTÈRES (numérique) ENCARACTÈRES (conteneur)

Cette fonction retourne un conteneur contenant chaque caractère du paramètre donné. La conversion en caractères du paramètre est effectuée en fonction du type du paramètre.

Si le paramètre est une [chaîne de caractères](#), le conteneur retourné contient comme éléments les caractères de la chaîne.

Si le paramètre est une [valeur numérique](#), celle-ci est convertie en chaîne de caractères avant d'effectuer la séparation en caractères.

Si le paramètre est un conteneur, la fonction **ENCARACTÈRES** convertit chaque élément en caractères individuels, regroupant tous les caractères obtenus dans un conteneur. Si le paramètre est un conteneur multidimensionnel, l'extraction de caractères est appliquée récursivement.

Exemples

```
SÉPARATEUR " , "
ÉCRIRE ENCARACTÈRES ("Bravo!")
ÉCRIRE ENCARACTÈRES (132.4)
ÉCRIRE ENCARAC ([1, ["Bye", 3], 4])
```

Pseudo-code C-9 : Exemples d'invocation de ENCARACTÈRES

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
[B,r,a,v,o,!]
[1,3,2,.,4]
[1,B,y,e,3,4]
```

Figure C-9 : Résultats de l'invocation de ENCARACTÈRES

Notez que chaque élément des conteneurs ci-dessus est un caractère. Ainsi, le conteneur **[1,B,y,e,3,4]** est en fait constitué des éléments '1', 'B', 'y', 'e', '3' et '4'.

ENCHAÎNE

Nom :	ENCHAÎNE
Synonymes :	Aucun
Type de retour :	Chaîne de caractères
Nombre d'arguments :	1
Description :	ENCHAÎNE convertit le paramètre en une chaîne de caractères.
Format(s) d'appel :	ENCHAÎNE (numérique) ENCHAÎNE (conteneur)

Cette fonction retourne une chaîne de caractères contenant le paramètre tel que représenté par une opération d'écriture à la console. Le paramètre peut être un [numérique](#), un [conteneur](#) ou même une chaîne de caractères (dans lequel cas aucune conversion n'est effectuée).

La fonction [FORMATER](#) peut être employée pour convertir plus d'un argument ou pour appliquer des formats de conversion.

Exemples

```
ÉCRIRE ENCHAÎNE ("Bravo!")
ÉCRIRE ENCHAÎNE (132.4)
ÉCRIRE ENCHAÎNE ([1, ["Bye", 3], 4])
```

Pseudo-code C-10 : Exemples d'invocation de ENCHAÎNE

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
Bravo!
132.4
1Bye34
```

Figure C-10 : Résultats de l'invocation de ENCHAÎNE

Notez dans l'exemple ci-dessus que l'invocation **ENCHAÎNE(132.4)** retourne la chaîne "132.4".

EXP

Nom :	EXP
Synonymes :	Aucun
Type de retour :	Numérique flottant
Nombre d'arguments :	0
Description :	EXP retourne la base du logarithme naturel.
Format(s) d'appel :	EXP

Cette fonction retourne e (2.71828182845905), la base du logarithme naturel (voir la fonction [LOGE](#)).

Exemples

```
ÉCRIRE EXP
ÉCRIRE LOGE (EXP)
```

Pseudo-code C-11 : Exemples d'invocation de EXP

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
2.71828182845905
1
```

Figure C-11 : Résultats de l'invocation de EXP

FINDECONTENU

Nom :	FINDECONTENU
Synonymes :	FDC
Type de retour :	Vrai ou faux
Nombre d'arguments :	1
Description :	FINDECONTENU retourne vrai si le canal d'entrées/sorties fourni a atteint la fin du contenu du fichier lu ou tampon lu.
Format(s) d'appel :	FINDECONTENU (<i>canal d'entrées/sorties</i>)

La fonction prédéfinie **FINDECONTENU** appliquée à un canal d'entrées/sorties permet de déterminer dans une [condition](#) si la fin du contenu d'un document lu (fichier ou tampon d'entrées/sorties) est atteinte.

La fonction n'est applicable qu'aux canaux d'entrées/sorties accessibles en [mode lecture](#). Tout appel à cette fonction avec un canal d'entrées/sorties associé à un document ouvert en mode **ÉCRITURE** ou **AJOUT** cause l'arrêt d'exécution de l'algorithme.

La fonction **FINDECONTENU** n'est pas applicable au clavier puisqu'il est impossible d'associer ce dernier à un canal d'entrées/sorties.

Exemples

```
OUVRIR "DONNEES" SUR 3 EN LECTURE
RÉPÉTER
  LIRE Valeur DE 3
  ÉCRIRE Valeur
JUSQU'À FINDECONTENU(3) OU Valeur < 0

SI FDC(3) ALORS
  ÉCRIRE "Toutes les données lues"
FINSI
```

Pseudo-code C-12 : Exemples d'invocation de FINDECONTENU

FORMATER

Nom :	FORMATER
Synonymes :	Aucun
Type de retour :	Chaîne de caractères
Nombre d'arguments :	1 ou plus
Description :	FORMATER retourne une chaîne de caractères constituée à partir d'une chaîne de formatage et d'une séquence optionnelle d'arguments.
Format(s) d'appel :	FORMATER (chaîne de formatage, séquence d'arguments)

Cette fonction formate une séquence d'arguments selon des directives de formatage fournies via la chaîne de formatage. On peut ainsi utiliser cette fonction pour formater des résultats avant de les afficher. La valeur retournée est une chaîne de caractères contenant les arguments formatés.

La fonction **ENCHAÎNE** peut être employée pour convertir un seul argument en chaîne de caractères, sans spécification de formatage.

Chaîne de formatage

Une chaîne de formatage contient deux types d'éléments : des caractères conventionnels et des *spécificateurs de format*. Les caractères conventionnels sont copiés tels quels dans la chaîne retournée. Les spécificateurs de format extraient les valeurs de la séquence d'arguments et y appliquent des directives de formatage.

Les spécificateurs de format ont la structure suivante (les crochets [et] indiquent un champs optionnel et ne font pas partie de la syntaxe des spécificateurs de format) :

%[**-**][*largeur*][*.précision*]*type*

Un spécificateur de format débute par le caractère %. Les éléments suivants font suite au % :

- Un indicateur de justification à gauche optionnel, **-**
- Un spécificateur de largeur optionnel, **largeur**
- Un spécificateur de précision optionnel, **.précision**
- Un spécificateur de conversion de type, **type**

Le tableau suivant résume les valeurs possibles de *type* et l'interprétation correspondante des éléments de formatage.

Type	Signification	Description
d	Décimal	L'argument doit être une valeur entière (LARP tente de convertir toute valeur d'un type autre à un entier). La valeur est convertie en chaîne de chiffres. Si le spécificateur de format comprend un spécificateur de précision, celui-ci indique que la chaîne résultante doit contenir au minimum le nombre indiqué de chiffres, des « 0 » étant insérés à gauche au besoin.

Type	Signification	Description
e	Scientifique	L'argument doit être une valeur flottante (<i>LARP</i> tente de convertir toute valeur d'un type autre à un flottant). La valeur est convertie en chaîne de la forme « d.ddd...E+ddd ». La chaîne résultante débute par le signe négatif si la valeur est négative. Un chiffre précède toujours le point décimal. Le nombre total de chiffres dans la chaîne résultante (incluant ceux précédant le point décimal) est donné via le spécificateur de précision, une précision par défaut de 15 chiffres étant appliquée. Le caractère d'exposant E dans la chaîne résultante est toujours suivi d'un signe + ou - et d'au moins trois chiffres.
f	Fixe	L'argument doit être une valeur flottante (<i>LARP</i> tente de convertir toute valeur d'un type autre à un flottant). La valeur est convertie en chaîne de la forme « d.ddd ». La chaîne résultante débute par le signe négatif si la valeur est négative. Un chiffre précède toujours le point décimal, un « 0 » étant affiché au besoin. Le nombre total de chiffres après le point décimal est donné via le spécificateur de précision, une précision par défaut de 2 chiffres étant appliquée.
g	Général	L'argument doit être une valeur flottante (<i>LARP</i> tente de convertir toute valeur d'un type autre à un flottant). La valeur est convertie en chaîne de la plus petite longueur possible selon la notation fixe (f) ou scientifique (e). Le nombre total de chiffres dans la chaîne résultante (incluant ceux précédant le point décimal) est donné via le spécificateur de précision, une précision par défaut de 15 chiffres étant appliquée. Les zéros inutiles sont éliminés de la fin de la chaîne résultante, et le point décimal n'est affiché qu'en cas de nécessité. La chaîne résultante exploite la notation fixe (f) si le nombre de chiffres à gauche du point décimal est moindre ou égal au spécificateur de précision, et si la valeur est supérieure ou égale à 0.00001. Sinon la chaîne résultante exploite la notation scientifique (e).
n	Nombre	L'argument doit être une valeur flottante (<i>LARP</i> tente de convertir toute valeur d'un type autre à un flottant). La valeur est convertie en chaîne de la forme « d,ddd,ddd.ddd ». Le format n correspond au format f avec l'ajout de séparateurs de milliers (généralement la virgule).
m	Montant d'argent	L'argument doit être une valeur flottante (<i>LARP</i> tente de convertir toute valeur d'un type autre à un flottant). La valeur est convertie en chaîne représentant un montant d'argent. La conversion est contrôlée par la configuration <i>Windows</i> [®] et ajustable via les <i>Options régionales</i> du <i>Panneau de configuration</i> . Si un spécificateur de précision est donné, ce dernier détermine le nombre de chiffres affichés après le point décimal (par défaut, 2 chiffres sont affichés).
s	Chaîne	L'argument doit être une chaîne de caractères (<i>LARP</i> tente de convertir toute valeur d'un type autre à une chaîne). Cette chaîne est insérée à la place du spécificateur de format. Le spécificateur de précision, si donné, indique la taille maximale de la chaîne résultante. Si la chaîne donnée en argument est plus longue, elle est tronquée.

Type	Signification	Description
x	Hexadécimal	L'argument doit être une valeur entière (LARP tente de convertir toute valeur d'un type autre à un entier). Cette valeur est convertie en une chaîne de chiffres hexadécimaux (base 16). Si le spécificateur de format comprend un spécificateur de précision, celui-ci indique que la chaîne résultante doit contenir au minimum le nombre indiqué de chiffres, des « 0 » étant insérés à gauche au besoin.

Puisque le caractère % indique le début d'un spécificateur de format, il ne peut pas être employé directement pour insérer le caractère % dans la chaîne résultante. Pour palier à cette restriction, le spécificateur %% représente le caractère % dans la chaîne de formatage. Ainsi, toute occurrence de %% dans la chaîne de formatage est remplacée par un seul % dans la chaîne résultante.

Les spécificateurs de conversion de type peuvent être donnés en lettres majuscules ou minuscules, sans distinction. Dans tous les spécificateurs de format pour arguments flottants, les caractères utilisés pour représenter le point décimal et le séparateur de milliers sont contrôlés via la configuration *Windows*[®] (voir *Options régionales* du *Panneau de configuration* de *Windows*[®]).

Les spécificateurs de *largeur* et de *précision* doivent être donnés sous forme décimale (exemple: %8.3f). Un spécificateur de *largeur* indique la largeur minimale du champs de conversion. Si la chaîne résultante est de longueur inférieure à la largeur spécifiée, des caractères blancs (i.e. des espaces) sont joints à la chaîne afin de l'allonger à la largeur prescrite. Par défaut, cette justification est vers la droite (i.e. les caractères blancs sont insérés avant la chaîne en argument); cependant si le spécificateur de justification à gauche (-) est compris dans le spécificateur de format, la justification est effectuée vers la gauche (i.e. les caractères blancs sont ajoutés après la chaîne en argument).

Exemples

Considérez les instructions suivantes faisant appel à l'instruction **FORMATER** afin de convertir diverses valeurs en chaînes de caractères :

```

\\ Exemples à un argument
ÉCRIRE '.....'
ÉCRIRE FORMATER(' 1. ***%8d***', 999)
ÉCRIRE FORMATER(' 2. ***%8.7d***', 999)
ÉCRIRE FORMATER(' 3. ***%-8d***', 999)
ÉCRIRE FORMATER(' 4. ***%e***', -999.99)
ÉCRIRE FORMATER(' 5. ***%14.5e***', -999.99)
ÉCRIRE FORMATER(' 6. ***%f***', -999.0)
ÉCRIRE FORMATER(' 7. ***%f***', -999.99)
ÉCRIRE FORMATER(' 8. ***%14.5f***', -999.99)
ÉCRIRE FORMATER(' 9. ***%g***', -999.0)
ÉCRIRE FORMATER('10. ***%g***', -999.99)
ÉCRIRE FORMATER('11. ***%14.5g***', -999.99)
ÉCRIRE FORMATER('12. ***%8n***', 99999.99)
ÉCRIRE FORMATER('13. ***%8m***', 99999.99)
ÉCRIRE FORMATER('14. ***%s***', "Allo")
ÉCRIRE FORMATER('15. ***%10s***', "Allo")
ÉCRIRE FORMATER('16. ***%-10s***', "Allo")
ÉCRIRE FORMATER('17. ***%x***', 123)

\\ Exemple avec plusieurs arguments
ÉCRIRE FORMATER("\n18. V=%d \n19. %s(V)=%8.4f", V, "Sin", $
Sinus(V-4.5))

```

Pseudo-code C-13 : Exemples d'invocation de FORMATER

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```

.....
1. ***      999***
2. *** 0000999***
3. ***999      ***
4. ***-9.9999000000000000E+002***
5. ***  -9.9999E+002***
6. ***-999***
7. ***-999.99***
8. ***      -999.99000***
9. ***-999***
10. ***-999.99***
11. ***      -999.99***
12. ***99,999.99***
13. ***$99,999.99***
14. ***Allo***
15. ***      Allo***
16. ***Allo      ***
17. ***7B***

18. V=10
19. Sin(V) = -0.7055

```

Figure C-12 : Résultats de l'invocation de FORMATER

HEURE

Nom :	HEURE
Synonymes :	Aucun
Type de retour :	Conteneur
Nombre d'arguments :	0
Description :	HEURE retourne un conteneur de quatre valeurs donnant l'heure courante : les heures, les minutes, les secondes et les millièmes de seconde.
Format(s) d'appel :	HEURE

Cette fonction retourne l'heure courante dans un conteneur. Les quatre valeurs contenues dans le conteneur retourné sont :

HEURE[1] sont les heures écoulées depuis le début du jour (0 à 23),

HEURE[2] sont les minutes écoulées depuis le début de l'heure (0 à 59),

HEURE[3] sont les secondes écoulées depuis le début de la minute (0 à 59), et

HEURE[4] sont les millièmes de secondes écoulés depuis le début de la seconde (0 à 999).

Exemples

```
ÉCRIRE HEURE
ÉCRIRE "Heure =", HEURE[1]
```

Pseudo-code C-14 : Exemples d'invocation de HEURE

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
[15 23 54 921]
15
```

Figure C-13 : Résultats de l'invocation de HEURE

LOG10

Nom :	LOG10
Synonymes :	Aucun
Type de retour :	Numérique flottant
Nombre d'arguments :	1
Description :	LOG10 retourne le logarithme en base 10 de la valeur donnée en paramètre.
Format(s) d'appel :	LOG10(numérique)

Cette fonction retourne le logarithme en base 10 (i.e. Log_{10}) de la valeur donnée en paramètre.

Si le paramètre fourni est une [chaîne de caractères](#), il y a tentative de conversion de celle-ci en valeur numérique.

Exemples

```
ÉCRIRE LOG10 (100)
ÉCRIRE LOG10 ("5.5")
```

Pseudo-code C-15 : Exemples d'invocation de LOG10

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
2
0.740362689494244
```

Figure C-14 : Résultats de l'invocation de LOG10

LOGE

Nom :	LOGE
Synonymes :	Aucun
Type de retour :	Numérique flottant
Nombre d'arguments :	1
Description :	LOGE retourne le logarithme en base e de la valeur donnée en paramètre.
Format(s) d'appel :	LOGE(<i>numérique</i>)

Cette fonction retourne le logarithme naturel (i.e. base *e*) de la valeur donnée en paramètre.

Si le paramètre fourni est une [chaîne de caractères](#), il y a tentative de conversion de celle-ci en valeur numérique.

La base du logarithme naturel peut être obtenue via la fonction [EXP](#).

Exemples

```
ÉCRIRE LOGE (100)
ÉCRIRE LOGE (EXP)
```

Pseudo-code C-16 : Exemples d'invocation de LOGE

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
4.60517018598809
1
```

Figure C-15 : Résultats de l'invocation de LOGE

MAJUSCULES

Nom :	MAJUSCULES
Synonymes :	Aucun
Type de retour :	Chaîne de caractères
Nombre d'arguments :	1
Description :	MAJUSCULES retourne la chaîne donnée en paramètre avec ses lettres minuscules converties en majuscules.
Format(s) d'appel :	MAJUSCULES (chaîne de caractères)

Cette fonction retourne la chaîne donnée en paramètre avec ses lettres minuscules converties en majuscules. Tout caractère n'étant pas une lettre minuscule demeure inchangé.

La fonction prédéfinie **MINUSCULES** transforme en minuscules les lettres majuscules retrouvées dans la chaîne donnée.

Exemples

```
ÉCRIRE MAJUSCULES("Bonjour le Monde!")
ÉCRIRE MAJUSCULES("Joe 99")
```

Pseudo-code C-17 : Exemples d'invocation de MAJUSCULES

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
BONJOUR LE MONDE!
JOE 99
```

Figure C-16 : Résultats de l'invocation de MAJUSCULES

MAXIMUM

Nom :	MAXIMUM
Synonymes :	MAX
Type de retour :	Séquence de numériques et/ou conteneurs
Nombre d'arguments :	1 ou plus
Description :	MAXIMUM retourne la plus grande valeur parmi celles fournies en paramètres.
Format(s) d'appel :	MAXIMUM (argument, argument, argument, ...)

Cette fonction accepte un nombre variable de paramètres et retourne la plus grande valeur parmi ceux-ci. Si un paramètre est un conteneur, la fonction parcourt récursivement le conteneur afin d'y identifier la plus grande valeur numérique.

Si un paramètre fourni est une [chaîne de caractères](#), il y a tentative de conversion de celle-ci en valeur numérique.

La fonction prédéfinie **MINIMUM** retourne la plus petite valeur parmi celles fournies en paramètres.

Exemples

```
ÉCRIRE MAXIMUM(10, 100, 20)
ÉCRIRE MAX(10, [100, [20, 200], "50"], 150))
```

Pseudo-code C-18 : Exemples d'invocation de MAXIMUM

Les résultats affichés dans la console d'exécution lors de l'exécution des instructions ci-dessus sont :

```
100
200
```

Figure C-17 : Résultats de l'invocation de MAXIMUM

MINIMUM

Nom :	MINIMUM
Synonymes :	MIN
Type de retour :	Séquence de numériques et/ou conteneurs
Nombre d'arguments :	1 ou plus
Description :	MINIMUM retourne la plus petite valeur parmi celles fournies en paramètres.
Format(s) d'appel :	MINIMUM (<i>argument, argument, argument, ...</i>)

Cette fonction accepte un nombre variable de paramètres et retourne la plus petite valeur parmi ces paramètres. Si un paramètre est un conteneur, la fonction parcourt récursivement le conteneur afin d'y identifier la plus petite valeur numérique.

Si un paramètre fourni est une [chaîne de caractères](#), il y a tentative de conversion de celle-ci en valeur numérique.

La fonction prédéfinie **MAXIMUM** retourne la plus grande valeur parmi celles fournies en paramètres.

Exemples

```
ÉCRIRE MINIMUM(10, 100, 20)
ÉCRIRE MIN(40, [100, [20, "12"], 50], 150))
```

Pseudo-code C-19 : Exemples d'invocation de MINIMUM

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
10
12
```

Figure C-18 : Résultats de l'invocation de MINIMUM

MINUSCULES

Nom :	MINUSCULES
Synonymes :	Aucun
Type de retour :	Chaîne de caractères
Nombre d'arguments :	1
Description :	MINUSCULES retourne la chaîne donnée en paramètre avec ses lettres majuscules converties en minuscules.
Format(s) d'appel :	MINUSCULES (<i>chaîne de caractères</i>)

Cette fonction retourne la chaîne donnée en paramètre avec ses lettres majuscules converties en minuscules. Tout caractère n'étant pas une lettre majuscule demeure inchangé.

La fonction prédéfinie **MAJUSCULES** transforme en majuscules les lettres minuscules retrouvées dans la chaîne donnée.

Exemples

```
ÉCRIRE MINUSCULES("Bonjour le Monde!")
ÉCRIRE MINUSCULES("Joe 99")
```

Pseudo-code C-20 : Exemples d'invocation de MINUSCULES

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
bonjour le monde!
joe 99
```

Figure C-19 : Résultats de l'invocation de MINUSCULES

PI

Nom :	PI
Synonymes :	Aucun
Type de retour :	Numérique flottant
Nombre d'arguments :	0
Description :	PI retourne la valeur de la constante trigonométrique π .
Format(s) d'appel :	PI

Cette fonction retourne la valeur de la constante trigonométrique π , en radians (3.14159265358979).

Exemples

<pre>ÉCRIRE PI ÉCRIRE SINUS (PI) + COSINUS (PI)</pre>

Pseudo-code C-21 : Exemples d'invocation de PI

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

<pre>3.14159265358979 -1</pre>

Figure C-20 : Résultats de l'invocation de PI

PLAFOND

Nom :	PLAFOND
Synonymes :	PLAF
Type de retour :	Numérique entier
Nombre d'arguments :	1
Description :	PLAFOND retourne le plus petit entier supérieur ou égal à la valeur donnée.
Format(s) d'appel :	PLAFOND (numérique)

Cette fonction retourne le plus petit entier supérieur ou égal à son paramètre. Si le paramètre est déjà entier, il est simplement retourné.

Si le paramètre fourni est une [chaîne de caractères](#), il y a tentative de conversion de celle-ci en valeur numérique.

Utilisez la fonction prédéfinie **PLANCHER** pour obtenir le *plus grand entier inférieur ou égal* à la valeur donnée.

Exemples

```
ÉCRIRE PLAFOND (11.32)
ÉCRIRE PLAF (" -1.5 ")
```

Pseudo-code C-22 : Exemples d'invocation de PLAFOND

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
12
-1
```

Figure C-21 : Résultats de l'invocation de PLAFOND

PLANCHER

Nom :	PLANCHER
Synonymes :	PLAN
Type de retour :	Numérique entier
Nombre d'arguments :	1
Description :	PLANCHER retourne le plus grand entier inférieur ou égal à la valeur donnée.
Format(s) d'appel :	PLANCHER (numérique)

Cette fonction retourne le plus grand entier inférieur ou égal à son paramètre. Si le paramètre est déjà entier, il est simplement retourné.

Si le paramètre fourni est une [chaîne de caractères](#), il y a tentative de conversion de celle-ci en valeur numérique.

Utilisez la fonction prédéfinie **PLAFOND** pour obtenir le *plus petit entier supérieur ou égal* à la valeur donnée.

Exemples

```
ÉCRIRE PLANCHER (11.32)
ÉCRIRE PLAN (" -1.5 ")
```

Pseudo-code C-23 : Exemples d'invocation de PLANCHER

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
11
-2
```

Figure C-22 : Résultats de l'invocation de PLANCHER

POSITION

Nom :	POSITION
Synonymes :	POS
Type de retour :	Numérique entier
Nombre d'arguments :	2
Description :	POSITION retourne la position d'une chaîne de caractères dans une autre chaîne de caractères, ou la position d'un élément dans un conteneur.
Format(s) d'appel :	POSITION (chaîne de caractères, chaîne de caractères) POSITION (élément, conteneur)

Cette fonction peut être utilisée pour rechercher la position (i.e. l'index) d'un élément dans une chaîne de caractère ou dans un conteneur. La recherche dépend du second paramètre :

- Si le second paramètre est une chaîne de caractère, le premier paramètre doit aussi être une chaîne de caractères ou un seul caractère (sinon, il y a tentative de conversion du premier paramètre en chaîne de caractères). La fonction retourne alors la position de la première occurrence de la première chaîne dans la seconde (la position retournée étant basée sur 1 comme index de départ). Si la première chaîne n'est pas retrouvée dans la seconde, la position 0 est retournée.
- Si le second paramètre est un conteneur, le premier paramètre peut alors être de n'importe quel type. La fonction recherche alors la première occurrence du premier paramètre dans le second (i.e. le conteneur). Si l'élément recherché n'est pas retrouvé dans le conteneur, la position 0 est retournée.

Notez que dans le cas d'une recherche dans un conteneur, la recherche n'est pas récursive (i.e. seule la première dimension d'un conteneur multidimensionnel est recherchée).

Exemples

```
ÉCRIRE POSITION("le", "Bonjour le monde!")
ÉCRIRE POSITION(3, [1, 2, [3, 4], 3, 5])
ÉCRIRE POSITION(4, [1, 2, [3, 4], 3, 5])
ÉCRIRE POS([3, 4], [1, 2, [3, 4], 3, 5])
```

Pseudo-code C-24 : Exemples d'invocation de POSITION

Les résultats affichés dans la console d'exécution lors de l'exécution des instructions ci-dessus sont :

```
9
4
0
3
```

Figure C-23 : Résultats de l'invocation de POSITION

RACINE

Nom :	RACINE
Synonymes :	Aucun
Type de retour :	Numérique flottant
Nombre d'arguments :	1
Description :	RACINE retourne la racine carrée de la valeur donnée en paramètre.
Format(s) d'appel :	RACINE (numérique)

Cette fonction retourne la racine carrée de la valeur donnée en paramètre.

Si le paramètre fourni est une [chaîne de caractères](#), il y a tentative de conversion de celle-ci en valeur numérique.

Exemples

<pre>ÉCRIRE RACINE (100) ÉCRIRE RACINE ("5.5")</pre>
--

Pseudo-code C-25 : Exemples d'invocation de RACINE

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

<pre>10 2.34520787991171</pre>

Figure C-24 : Résultats de l'invocation de RACINE

SINUS

Nom :	SINUS
Synonymes :	SIN
Type de retour :	Numérique flottant
Nombre d'arguments :	1
Description :	SINUS retourne la valeur de la fonction trigonométrique <i>Sin</i> appliquée à l'angle donnée en paramètre (en radians).
Format(s) d'appel :	SINUS (numérique)

Cette fonction retourne le sinus (*Sin*) de la valeur numérique (angle en radians) fournie en paramètre. La valeur de retour est de type flottant.

Si le paramètre fourni est une [chaîne de caractères](#), il y a tentative de conversion de celle-ci en valeur numérique.

Exemples

```
ÉCRIRE SINUS (1.2)
ÉCRIRE SIN (1 - 2.7)
```

Pseudo-code C-26 : Exemples d'invocation de SINUS

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
0.932039085967226
-0.991664810452469
```

Figure C-25 : Résultats de l'invocation de SINUS

SOUSENSEMBLE

Nom :	SOUSENSEMBLE
Synonymes :	Aucun
Type de retour :	Chaîne de caractères ou conteneur
Nombre d'arguments :	3
Description :	SOUSENSEMBLE retourne un sous-ensemble du premier paramètre.
Format(s) d'appel :	SOUSENSEMBLE(<i>chaîne de caractères, début, longueur</i>) SOUSENSEMBLE(<i>conteneur, début, longueur</i>)

Cette fonction est utilisée pour obtenir une copie partielle du contenu du premier paramètre, qui doit être une chaîne de caractère ou un conteneur. Le second paramètre (*début*) indique l'index où commencer à extraire les caractères ou éléments du premier paramètre, et le troisième paramètre (*longueur*) indique le nombre de caractères (ou le nombre d'éléments) à copier. La valeur retournée dépend du premier paramètre :

- Si le premier paramètre est une chaîne de caractère, la fonction retourne une chaîne de caractères composée des *longueur* caractères du premier paramètre à partir de l'index *début*.
- Si le premier paramètre est un conteneur, la fonction retourne un conteneur composé des *longueur* éléments du premier paramètre à partir de l'index *début*.

Si un des deux derniers paramètres fournis est une chaîne de caractères, il y a tentative de conversion de celle-ci en valeur numérique.

Exemples

```
ÉCRIRE SOUSENSEMBLE("Bonjour le monde!", 12, 5)
ÉCRIRE SOUSENSEMBLE("Bonjour le monde!", 9, 50)
ÉCRIRE SOUSENSEMBLE([10, 20, 30, 40, 50], 2, "3")
```

Pseudo-code C-27 : Exemples d'invocation de SOUSENSEMBLE

Les résultats affichés dans la [console d'exécution](#) lors de l'exécution des instructions ci-dessus sont :

```
monde
le monde!
[20 30 40]
```

Figure C-26 : Résultats de l'invocation de SOUSENSEMBLE

Annexe D - Éléments de syntaxe

Cette section résume les principaux éléments de la syntaxe de pseudo-code et d'organigramme LARP. Dans les tableaux ci-dessous, les mots réservés sont présentés en caractères gras et majuscules (par exemple, **DÉBUT**), les éléments à développer en caractères italiques (par exemple, *séquence d'instructions*) et les éléments optionnels entre crochets (par exemple, [*liste de paramètres*]).

Pour plus d'information sur un élément de syntaxe en particulier, consultez la section correspondante.

D.1 Modules

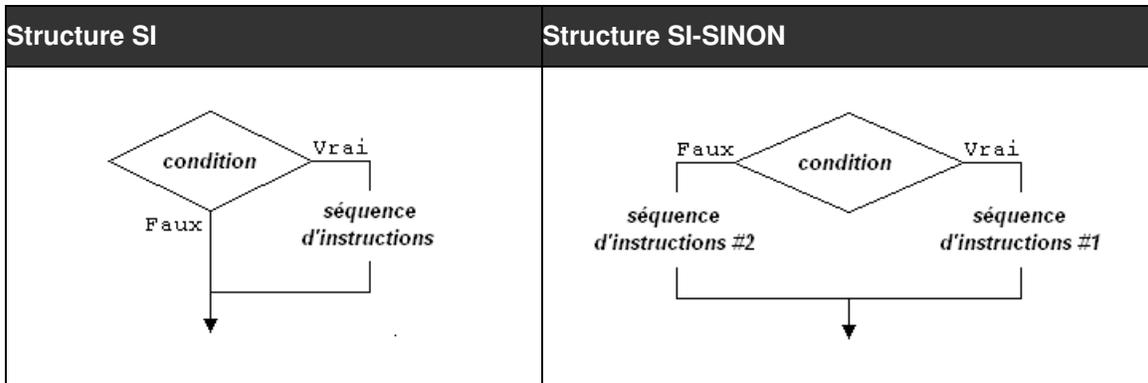
Module principal	Module auxiliaire
DÉBUT <i>séquence d'instructions</i> FIN	ENTRER [<i>liste de paramètres</i>] <i>séquence d'instructions</i> RETOURNER [<i>valeur de retour</i>]

où

- ***séquence d'instructions*** est une séquence d'instructions LARP autres qu'une [définition de module](#) (les définitions de modules ne peuvent pas être imbriquées).
- ***liste de paramètres***, optionnelle, est une ou plusieurs [variables](#) séparées par des virgules.
- ***valeur de retour***, optionnelle, est une expression qui retourne une [valeur entière](#), une [valeur flottante](#), une [chaîne de caractères](#) ou un [conteneur](#).

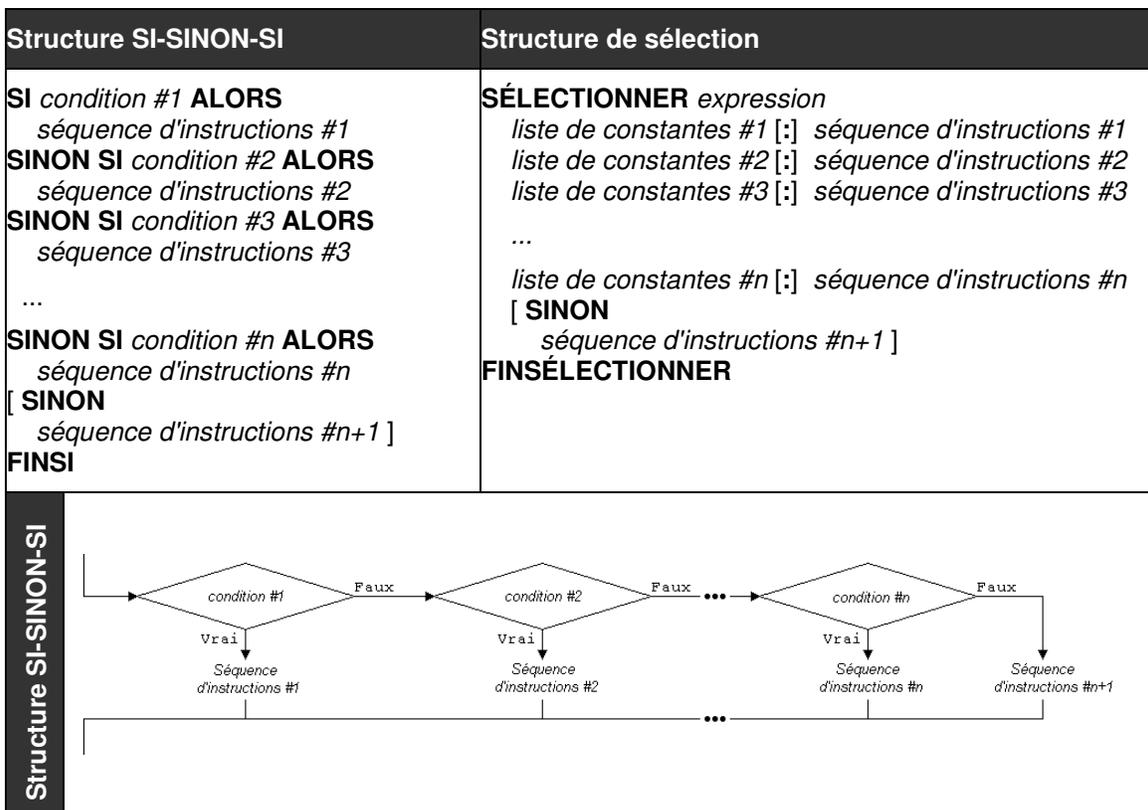
D.2 Structures conditionnelles

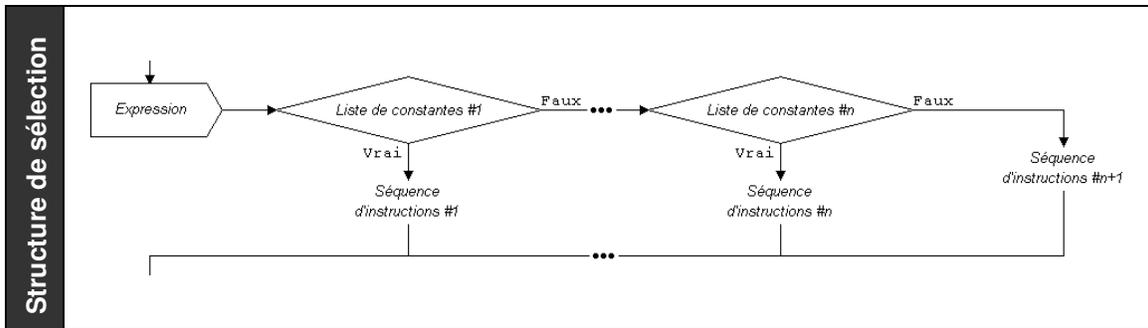
Structure SI	Structure SI-SINON
SI <i>condition</i> ALORS <i>séquence d'instructions</i> FINSI	SI <i>condition</i> ALORS <i>séquence d'instructions #1</i> SINON <i>séquence d'instructions #2</i> FINSI



où

- **condition** est une expression booléenne composée d'opérateurs relationnels, d'opérateurs logiques et de tests de type.
- **séquence d'instructions #** sont des séquences d'instructions LARP autres qu'une définition de module (les définitions de modules ne peuvent pas être imbriquées).





où

- **condition** est une expression booléenne composée d'opérateurs relationnels, d'opérateurs logiques et de tests de type.
- **expression** est une expression retournant une valeur entière, une valeur flottante, une chaîne de caractères ou un conteneur.
- **séquence d'instructions #** sont des séquences d'instructions LARP autres qu'une définition de module (les définitions de modules ne peuvent pas être imbriquées).
- **liste de constantes** sont des énumérations de valeurs entières, valeurs flottantes, chaînes de caractères et/ou conteneurs, séparées par des virgules. Optionnellement, deux points (:) peuvent être insérés entre une liste de constantes et sa séquence d'instructions.
- La dernière section, commençant par **SINON**, est optionnelle.

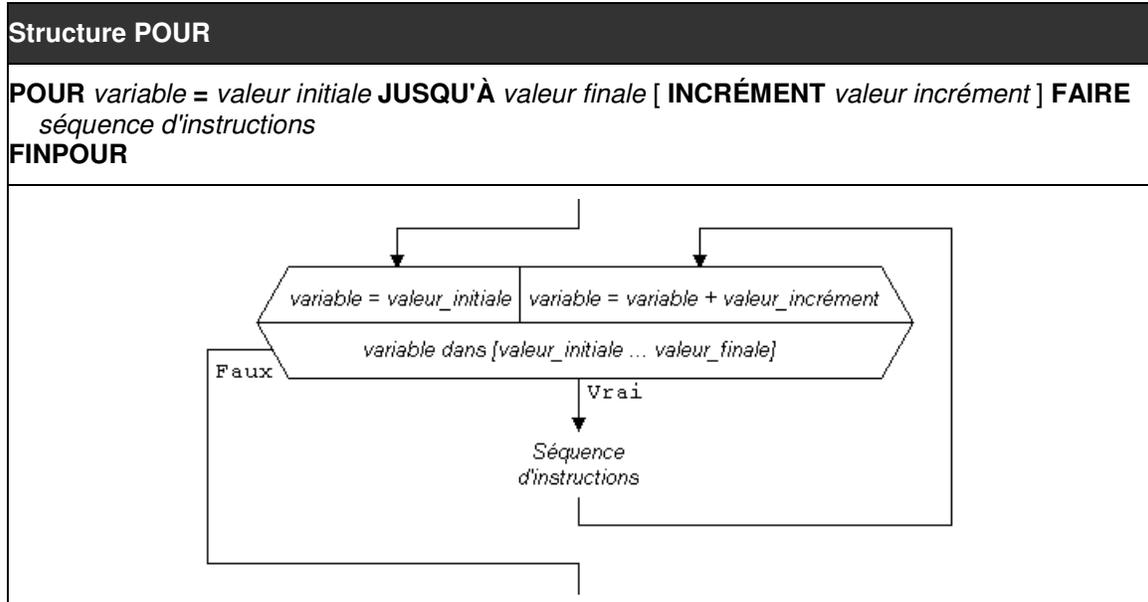
D.3 Structures répétitives

Structure TANTQUE	Structure RÉPÉTER-JUSQU'À
TANTQUE <i>condition</i> FAIRE <i>séquence d'instructions</i> FINTANTQUE	RÉPÉTER <i>séquence d'instructions</i> JUSQU'À <i>condition</i>

où

- **condition** est une expression booléenne composée d'opérateurs relationnels, d'opérateurs logiques et de tests de type.

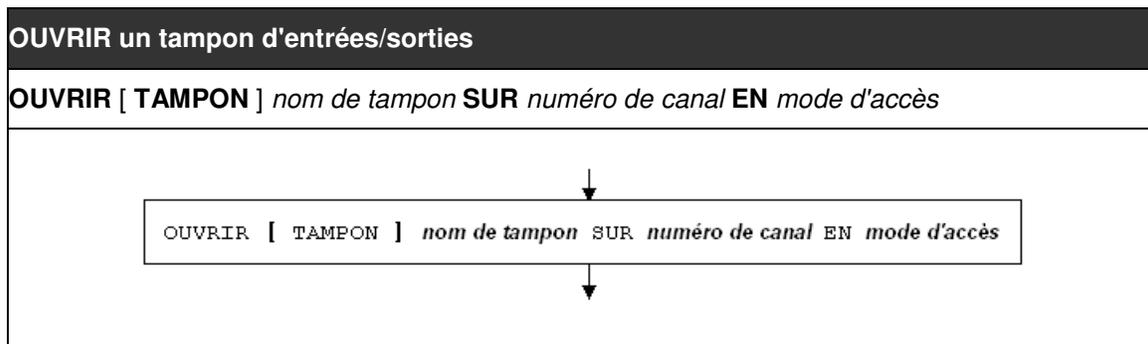
- **séquence d'instructions** est une séquence d'instructions *LARP* autres qu'une [définition de module](#) (les définitions de modules ne peuvent pas être imbriquées).



où

- **variable** est un [nom de variable](#).
- **valeur initiale**, **valeur finale** et **valeur incrément** (optionnelle) sont des expressions retournant une [valeur entière](#).
- **séquence d'instructions** est une séquence d'instructions *LARP* autres qu'une [définition de module](#) (les définitions de modules ne peuvent pas être imbriquées).

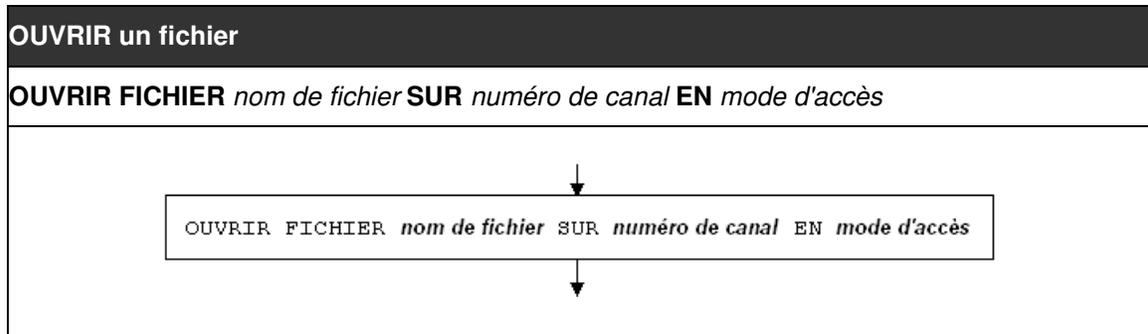
D.4 Fichiers et tampons d'entrées/sorties



où

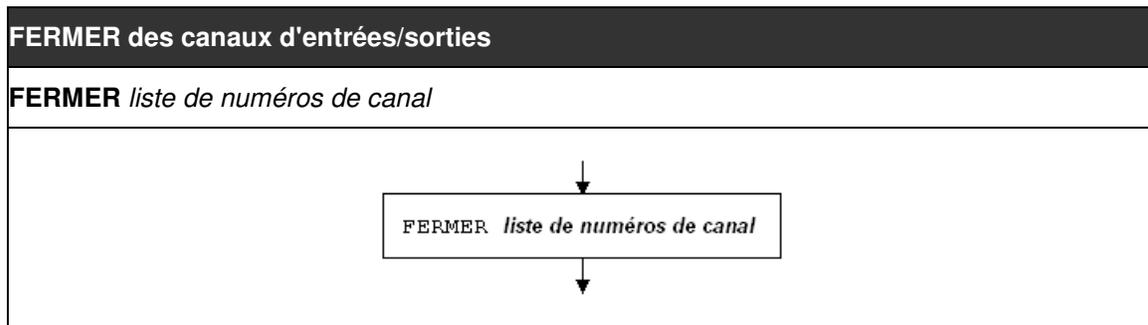
- **nom de tampon** est le nom du [tampon d'entrées/sorties](#) à ouvrir (le tampon doit être créé au préalable et apparaître dans le [navigateur de documents](#)).
- **numéro de canal** est une expression retournant une [valeur entière](#) de 1 à 256.

- **mode d'accès** est un des mots réservés suivants: **LECTURE**, **ÉCRITURE** ou **AJOUT**.
- La mot réservé **TAMPON** est optionnel.



où

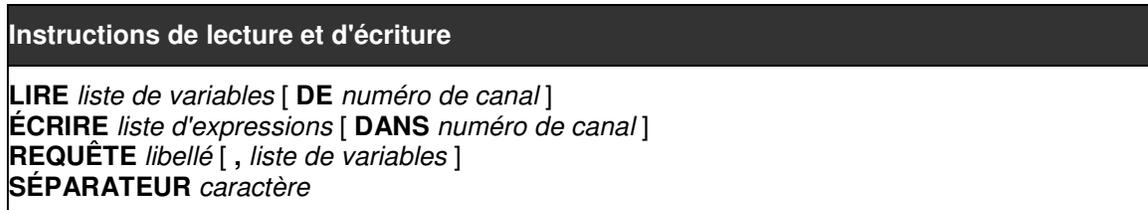
- **nom de fichier** est le nom du **fichier** à ouvrir.
- **numéro de canal** est une expression retournant une **valeur entière** de 1 à 256.
- **mode d'accès** est un des mots réservés suivants: **LECTURE**, **ÉCRITURE** ou **AJOUT**.

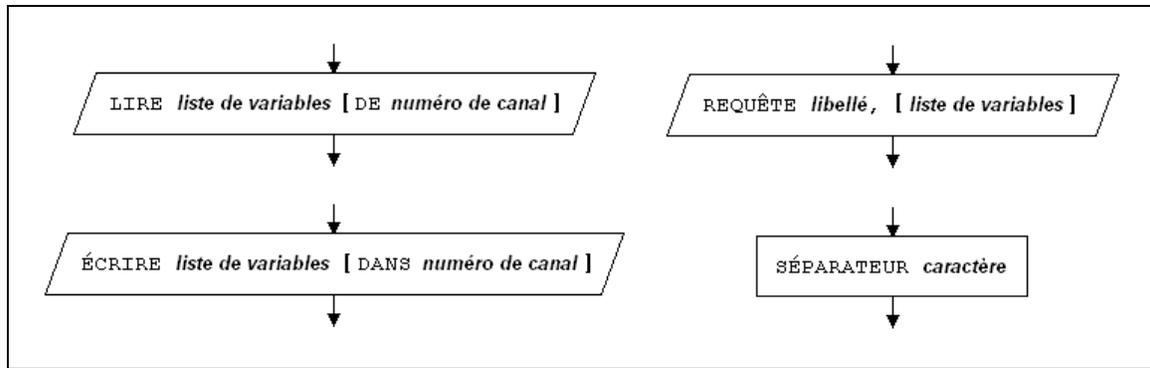


où

- **liste de numéros de canal** est une liste d'expressions séparées par des virgules, chacune retournant une **valeur entière** de 1 à 256.

D.5 Lecture et écriture





où

- **liste de variables** est constitué d'une ou plusieurs **variables** séparées par des virgules.
- **numéro de canal**, optionnel, est une expression retournant une **valeur entière** de 1 à 256.
- **liste d'expressions** est une liste d'une ou plusieurs expressions retournant chacune une **valeur entière**, une **valeur flottante**, une **chaîne de caractères** ou un **conteneur**.
- **libellé** est une expression retournant une **chaîne de caractères**.
- **caractère** est une expression retournant une **chaîne de caractères** contenant un seul caractère.
- si aucun **numéro de canal** n'est fourni, l'entrée et la sortie se font via la **console d'exécution**.

Annexe E - Avertissements et erreurs

Les messages affichés par *LARP* sont classés en deux catégories :

1. **Les messages reliés à l'environnement de développement** : messages affichés suite à des actions ou commandes inappropriées de la part de l'utilisateur.
2. **Les messages reliés à l'exécution d'algorithmes** : messages affichés suite à des erreurs de syntaxes dans les modules ou à une opération invalide effectuée lors de l'exécution d'un algorithme.

E.1 Messages reliés à l'environnement de développement

Les messages qui suivent sont généralement affichés de façon ponctuelle (dans une fenêtre informative) lorsque l'[environnement de développement](#) refuse une opération de l'utilisateur ou lorsque plus d'information sont requises afin d'effectuer l'opération demandée.

Les messages sont classés par ordre alphabétique, accompagnés chacun d'une brève description.

Aucune aide n'est disponible pour ce type d'erreur.

L'[aide en ligne](#) de *LARP* ne comporte aucune information supplémentaire sur ce type d'erreur.

Clé SparKey introuvable.

Une erreur s'est produite lors de la détection et/ou la vérification de la [clé de débridage](#) présentement branchée à l'ordinateur. Assurez-vous que la clé de débridage est bien branchée, que votre [pseudonyme](#) correspond à celui de la clé et que la clé n'est pas endommagée. Si tout semble correct, contactez le [support technique](#) afin de remplacer votre clé de débridage.

Codes d'accès primaires invalides.

Le [pseudonyme](#) fourni pour activer le [mode super-utilisateur](#) ne correspond pas à celui de la [clé de débridage](#) présentement branchée à l'ordinateur. Assurez-vous que la clé de débridage est bien branchée et que votre pseudonyme correspond à celui de la clé.

Configuration de clé invalide.

La [configuration de la clé de débridage](#) présentement branchée à l'ordinateur est erronée. Cette erreur est généralement causée par une clé de débridage endommagée ou d'origine autre que celles fournies par le fournisseur de *LARP*. Contactez le [support technique](#) afin de remplacer votre clé de débridage.

Erreur à l'ouverture de clé SparKey.

Une erreur s'est produite lors de la détection et/ou la vérification de la [clé de débridage](#) présentement branchée à l'ordinateur. Assurez-vous que la clé de débridage est bien branchée, que votre [pseudonyme](#) correspond à celui de la clé et que la clé n'est pas endommagée. Si tout semble correct, contactez le [support technique](#) afin de remplacer votre clé de débridage.

Erreur de fermeture de clé SparKey.

Une erreur s'est produite lors de l'accès à la [clé de débridage](#) présentement branchée à l'ordinateur. Assurez-vous que la clé de débridage est bien branchée et que votre [pseudonyme](#) correspond à celui de la clé.

Erreur inconnue.

Une erreur non prévue par les développeurs de LARP fut détectée. Veuillez contacter le [support technique](#) de LARP afin d'obtenir de l'aide.

Erreur de lecture - nombre d'octets ou adresse invalide.

Une erreur s'est produite lors de l'accès à la [clé de débridage](#) présentement branchée à l'ordinateur. Assurez-vous que la clé de débridage est bien branchée, que votre [pseudonyme](#) correspond à celui de la clé et que la clé n'est pas endommagée. Si c'est le cas, contactez le [support technique](#) afin de remplacer votre clé de débridage.

Erreur de lecture de données.

Une erreur s'est produite lors de l'accès à la [clé de débridage](#) présentement branchée à l'ordinateur. Assurez-vous que la clé de débridage est bien branchée et que la clé n'est pas endommagée. Si c'est le cas, contactez le [support technique](#) afin de remplacer votre clé de débridage.

Erreur de lecture de données d'identification.

Une erreur s'est produite lors de l'accès à la [clé de débridage](#) présentement branchée à l'ordinateur. Assurez-vous que la clé de débridage est bien branchée, que votre [pseudonyme](#) correspond à celui de la clé et que la clé n'est pas endommagée. Si c'est le cas, contactez le [support technique](#) afin de remplacer votre clé de débridage.

Format du code d'accès invalide.

Le [pseudonyme](#) fourni pour activer le [mode super-utilisateur](#) est d'un format incompatible à celui de la [clé de débridage](#) présentement branchée à l'ordinateur. Assurez-vous que votre pseudonyme correspond à celui de la clé branchée à l'ordinateur.

«*Identificateur*» n'est pas un nom de module valide.

Le nom que vous désirez donner au module est invalide. Le nom d'un module doit être composé de lettres, de chiffres et/ou du caractère de soulignement (_), et le premier caractère du nom ne doit pas être un chiffre. Les espaces ne sont pas permis dans un nom de module.

Pour plus d'information, consultez la section portant sur les [modules](#).

«*Identificateur*» n'est pas un nom de tampon d'E/S valide.

Le nom que vous désirez donner au tampon d'entrées/sorties est invalide. Le nom d'un tampon doit être composé de lettres, de chiffres et/ou du caractère de soulignement (_), et le premier caractère du nom ne doit pas être un chiffre. Les espaces ne sont pas permis dans un nom de tampon d'entrées/sorties.

Pour plus d'information, consultez la section portant sur les [tampons d'entrées/sorties](#).

Le module principal d'un projet ne peut pas être supprimé.

Vous tentez de détruire le [module principal](#) du projet. Tout projet *LARP* doit disposer d'un module principal, car c'est le point de départ d'exécution de l'algorithme. Si vous désirez changer le module principal d'un projet, vous devez modifier le contenu du module principal existant.

Le projet ne contient aucun tampon d'entrées/sorties à imprimer.

L'impression de [modules](#) est restreint au mode [super-utilisateur](#). Cependant l'impression des [tampons d'entrées/sorties](#) est permise pour tout utilisateur. Vous tentez d'imprimer un projet sans tampon d'entrées/sorties alors que le mode [super-utilisateur](#) n'est pas activé.

Les fichiers d'aide ne sont pas accessibles.

Il semble que certains fichiers relatifs à [l'aide en ligne](#) de *LARP* soient manquants. Assurez-vous d'avoir bien installé *LARP* et que les fichiers d'aide sont installés. Si ce n'est pas le cas, réinstallez la plus récente version de *LARP*. Si la réinstallation ne corrige pas la situation, contactez le [support technique](#).

Pseudonyme invalide.

Le [pseudonyme](#) fourni lors du démarrage de *LARP* ou lors du changement d'utilisateur est de format invalide. Consultez la section portant sur le [mode super-utilisateur](#) pour plus d'information.

Un document nommé «*nom de document*» existe déjà dans le projet.

Vous tentez de créer un nouveau document ([module](#) ou [tampon d'entrées/sorties](#)) ayant le nom d'un autre document du projet. Sélectionnez un nom différent.

Notez que *LARP* peut avoir abrégé le nom de document que vous avez sélectionné en y éliminant les caractères non permis. Notez aussi que *LARP* ignore les accents des caractères accentués retrouvés dans un nom de document; ainsi les noms **DONNÉES** et **DONNEES** font référence à un même document, nommé **DONNEES**.

Un seul fichier à la fois peut être accepté.

L'[environnement de développement](#) de *LARP* accepte les fichiers déposés (i.e. par *drag-and-drop*), mais seulement un fichier à la fois.

Une copie de sûreté générée suite à un crash de *LARP* est disponible. Voulez-vous restaurer ce projet?

À intervalles régulières ainsi que lorsque l'algorithme est exécuté, une copie de sûreté du projet est automatiquement conservée dans un fichier temporaire de l'ordinateur. Cette copie de sûreté est automatiquement détruite lorsque le projet édité est sauvegardé ou lorsque l'exécution de l'algorithme est terminée.

Si une erreur fatale résultant en un crash de *LARP* se produit, la copie de sûreté n'est pas détruite. Au redémarrage de *LARP*, celui-ci détecte l'existence de la copie et propose à l'utilisateur de recharger ce projet dans l'éditeur. Ainsi, si *LARP* crash avant que vous aillez eu l'occasion de sauvegarder votre projet, vous pouvez le récupérer au prochain démarrage de *LARP*.

Consultez la section portant sur les [sauvegardes de sécurité](#) pour plus d'information.

Attention : si vous refusez de restaurer le projet lors du redémarrage de *LARP*, la copie de sûreté est irrémédiablement effacée.

Voulez-vous vraiment effacer les données ci-dessus associées au projet?

En tant que [super-utilisateur](#), vous avez la possibilité d'effacer les [statistiques](#) ainsi que le [pseudonyme](#) rattaché au projet. Une fois effacées, ces données ne sont plus récupérables.

Vous devez fermer le projet actuel avant de changer d'utilisateur.

Lors de l'exécution d'un algorithme, certaines commandes habituellement accessibles via l'[environnement de développement](#) sont temporairement désactivées jusqu'à la fermeture de la [console d'exécution](#), i.e. jusqu'à ce que l'exécution de l'algorithme soit terminée.

Vous devez obligatoirement fermer la console (soit en complétant ou en interrompant l'exécution de l'algorithme) avant de changer le [pseudonyme](#).

Vous devez au préalable fermer la console.

Lors de l'exécution d'un algorithme, certaines commandes habituellement accessibles via l'[environnement de développement](#) sont temporairement désactivées jusqu'à la fermeture de la [console d'exécution](#), i.e. jusqu'à ce que l'exécution de l'algorithme soit terminée.

Vous devez obligatoirement fermer la console (soit en complétant ou en interrompant l'exécution de l'algorithme) avant d'effectuer l'opération désirée.

E.2 Messages reliés à l'exécution d'algorithmes

Les messages ci-dessous sont affichés lorsqu'un problème est détecté durant la [compilation ou l'exécution](#) d'algorithmes. Ces messages sont toujours affichés dans le [panneau de messages](#) et, lorsque le problème est détecté à l'exécution d'un algorithme, dans une fenêtre ponctuelle.

Les messages d'avertissement indiquent généralement une erreur potentielle dans un algorithme, mais cette erreur n'étant pas fatale, l'exécution se poursuit.

Les messages d'erreur sont généralement affichés :

- **lors de la compilation d'un algorithme** : la compilation est tout de même complétée afin de valider la syntaxe du reste de l'algorithme, mais il est impossible d'exécuter l'algorithme.
- **lors de l'exécution d'un algorithme** : la plupart des erreurs causent généralement une interruption de l'exécution de l'algorithme.

Certains messages d'erreur identifient une erreur détectée dans le logiciel *LARP*. Ces bogues doivent être [rapportés](#) au support technique de *LARP* afin que le problème soit corrigé dans la prochaine version du logiciel.

E1001 Un module auxiliaire doit débiter par l'instruction ENTRER

Vous avez un [module auxiliaire](#) débutant avec une instruction autre que **ENTRER** (avec optionnellement des paramètres).

Si le module en erreur débute par l'instruction **DÉBUT**, rappelez-vous que seul le **module principal** peut commencer avec l'instruction **DÉBUT**.

Consultez la section portant sur les **modules** pour plus d'information.

E1002 Le module principal doit débiter par l'instruction DÉBUT

Le **module principal** du projet débute avec une instruction autre que **DÉBUT**.

Si le module en erreur débute par l'instruction **ENTRER**, rappelez-vous que seuls les **modules auxiliaires** peuvent commencer avec l'instruction **ENTRER**.

Consultez la section portant sur les **modules** pour plus d'information.

E1003 En-tête de module invalide

Le format de l'instruction débutant le module en erreur ne correspond pas à celui imposé par *LARP*.

Consultez la section portant sur les **modules** pour plus d'information.

E1004 Un module auxiliaire doit être terminé par une instruction RETOURNER

Vous avez un **module auxiliaire** dont la dernière instruction est autre que **RETOURNER** (avec optionnellement une valeur de retour).

Si le module en erreur se termine par l'instruction **FIN**, rappelez-vous que seul le **module principal** peut être terminés avec l'instruction **FIN**.

Consultez la section portant sur les **modules** pour plus d'information.

E1005 Le module principal doit être terminé par l'instruction FIN

Le **module principal** du projet se termine avec une instruction autre que **FIN**.

Si le module en erreur se termine par l'instruction **RETOURNER**, rappelez-vous que seuls les **modules auxiliaires** peuvent être terminés avec l'instruction **RETOURNER**.

Consultez la section portant sur les **modules** pour plus d'information.

E1006 Un module doit être terminé par RETOURNER ou FIN

Le module du projet n'est pas terminé par une instruction appropriée selon le type de module. Une instruction appropriée serait **FIN** ou **RETOURNER** (avec optionnellement une valeur de retour).

Consultez la section portant sur les **modules** pour plus d'information.

E1007 Les variables doivent être séparées par des virgules

Certaines instructions requièrent une liste de **variables** (par exemple, **LIRE**, **REQUÊTE** et **ENTRER**). Lorsque plus d'une variable sont spécifiées, celles-ci doivent être séparées par une virgule (,) dans la liste.

Par exemple, l'instruction **LIRE A B C** est invalide; elle devrait plutôt s'écrire **LIRE A, B, C**.

E1008 L'identificateur «*nom de variable*» est un nom de module

Vous tentez d'employer le nom d'un [module](#) comme [variable](#) (par exemple dans une [affectation](#)). Le nom employé correspond à un module du projet, mais ce module n'est pas invoqué de façon adéquate.

Consultez la section portant sur les [modules](#) pour plus d'information.

E1009 Le mot réservé «*mot réservé*» ne peut pas être employé dans ce contexte

Vous tentez d'employer un mot réservé du langage *LARP* dans un contexte autre que ceux auxquels s'applique l'instruction visée.

La cause de l'erreur peut être que vous tentiez d'employer un mot réservé comme [nom de variable](#).

E1010 Je ne comprend pas cet énoncé

Vous ne respectez pas la syntaxe du langage *LARP*, et il est impossible de fournir plus d'information sur l'erreur détectée.

Consultez l'[aide en ligne](#) correspondant à l'instruction que vous tentez d'exploiter et assurez-vous de bien respecter la syntaxe permise dans les pseudo-codes et organigrammes *LARP*.

E1011 Les champs de modèles doivent être remplacés par du pseudo-code valide

Vous avez glissé un modèle du [panneau de modèles](#) vers votre pseudo-code, mais vous avez oublié de remplacer les champs à compléter par du pseudo-code valide. Ces champs sont identifiés entre accolades (`{` et `}`) et doivent être remplacés (accolades comprises) par du pseudo-code valide.

Par exemple, lorsque vous insérez un modèle de lecture dans votre pseudo-code, la ligne **LIRE `{liste_variables}`** apparaît. Vous devez obligatoirement remplacer le champ **`{liste_variables}`** par une ou plusieurs variables.

E1012 La condition de la structure conditionnelle ou itérative correspondante est erronée

Une erreur s'est glissée dans la formulation de la [condition](#) d'une [structure conditionnelle](#) ou d'une [structure répétitive](#).

Veillez consulter les sections correspondantes pour plus d'information sur la formulation des conditions.

E1999 Erreur inconnue; contactez le support technique

Une erreur non prévue par les développeurs de *LARP* fut détectée.

Veillez contacter le [support technique](#) de *LARP* afin d'obtenir de l'aide.

E2001 Pas assez d'arguments fournis en appel de module

Vous faites appel à un [module auxiliaire](#) en fournissant un nombre insuffisant d'arguments. Le nombre d'arguments fournis doit évaluer le nombre de paramètres énumérés en en-tête du module invoqué.

Consultez la section portant sur les [modules paramétrés](#) pour plus d'information.

E2002 Trop d'arguments fournis en appel de module

Vous faites appel à un [module auxiliaire](#) en fournissant un nombre trop élevé d'arguments. Le nombre d'arguments fournis doit égaler le nombre de paramètres énumérés en en-tête du module invoqué.

Consultez la section portant sur les [modules paramétrés](#) pour plus d'information.

E2003 Valeur de type invalide

Une valeur numérique, une expression ou une variable contenant une valeur numérique d'un type inapproprié est exploitée dans une instruction.

Une cause fréquente de cette erreur est d'employer dans une instruction d'algorithme *LARP* une [variable](#) dont le contenu est inapproprié pour l'instruction (par exemple, fournir une valeur inappropriée comme argument à une [fonction prédéfinie](#)).

E2004 Index invalide

L'index fourni en référence à un élément de [conteneur](#) est invalide. Un index doit absolument être une valeur entière.

Par exemple, l'instruction **ÉCRIRE a[1.2]** résulte en une erreur car l'index spécifié n'est pas un entier.

Pour plus d'information, consultez la section portant sur l'[accès aux éléments de conteneurs](#).

E2005 Index hors limites

L'index fourni en référence à un élément de [conteneur](#) est de valeur inférieure à l'index minimale permis ou supérieure au nombre d'éléments dans le conteneur.

Pour plus d'information, consultez la section portant sur l'[accès aux éléments de conteneurs](#).

E2006 La référence n'est pas une chaîne de caractères

L'instruction exploitée requiert l'utilisation d'une [chaîne de caractères](#), mais la valeur fournie est d'un autre type (i.e. un nombre, un conteneur ou autres).

E2007 La référence n'est pas un conteneur

L'instruction exploitée requiert l'utilisation d'un [conteneur](#), mais la valeur fournie est d'un autre type (i.e. un nombre, une chaîne de caractères ou autres).

E2008 Impossible de dimensionner un conteneur

Le [conteneur](#) manipulé ne peut être dimensionné selon les besoins de l'instruction. Cette erreur est généralement causée par un manque de mémoire de l'ordinateur.

Pour éliminer ce problème, utilisez un conteneur aux dimensions moindres.

Pour configurer la taille maximale d'un conteneur manipulable dans un algorithme *LARP*, consulter la section portant sur la [configuration de la console d'exécution](#).

E2009 Le conteneur est vide

Le [conteneur](#) fourni contient aucune valeur.

E2010 Nom de fichier ou tampon d'E/S invalide

Le nom fourni à l'instruction **OUVRIR** est invalide.

Lors de l'**ouverture d'un fichier**, le nom fourni est invalide s'il n'est pas conforme aux noms de fichiers permis par *Windows*[®], ou si le fichier à ouvrir en lecture n'existe pas.

Lors de l'**ouverture d'un tampon d'entrées/sorties**, le nom de tampon fourni est invalide s'il correspond à aucun tampon du projet.

Consultez la section portant sur les **fichiers et tampons d'entrées/sorties** pour plus d'information.

E2011 Numéro de canal invalide

Le **numéro de canal** fourni à l'instruction **OUVRIR** est invalide. Les canaux d'entrées/sorties disponibles dans *LARP* sont numérotés de 1 à 256.

Consultez la section portant sur les **canaux d'entrées/sorties** pour plus d'information.

E2012 Canal déjà alloué à un autre fichier ou tampon d'E/S

Le **numéro de canal** fourni à l'instruction **OUVRIR** est déjà associé à un autre **fichier** ou **tampon d'entrées/sorties** ouvert. Il est interdit d'associer un même canal d'entrées/sorties simultanément à deux documents ouverts.

Consultez la section portant sur les **canaux d'entrées/sorties** pour plus d'information.

E2013 Accès à un canal non alloué

Le **numéro de canal** fourni à l'instruction n'est associé à aucun **fichier** ou **tampon d'entrées/sorties**.

Avant d'exploiter un canal d'entrées/sorties lors d'une lecture ou d'une écriture, le canal doit au préalable être associé à un fichier ou un tampon d'entrées/sorties via l'instruction **OUVRIR**.

Consultez la section portant sur les **canaux d'entrées/sorties** pour plus d'information.

E2014 Accès invalide au canal spécifié

Le **numéro de canal** fourni à l'instruction de **LIRE** ou **ÉCRIRE** ne permet pas l'opération désirée.

Lors de l'ouverture d'un **fichier** ou un **tampon d'entrées/sorties** via l'instruction **OUVRIR**, on doit spécifier le **mode d'accès** au document, i.e. si le document est ouvert afin d'y lire des données ou afin d'y écrire des résultats.

Cette erreur est détectée lorsqu'un algorithme tente de lire d'un document (via son canal) ouvert en mode écriture, ou d'écrire dans un document ouvert en mode lecture.

Consultez les sections portant sur les **canaux d'entrées/sorties** et l'**ouverture d'un canal d'entrées/sorties** pour plus d'information.

E2015 Fichier ou tampon d'E/S déjà ouvert sur un autre canal

Le nom du **fichier** ou du **tampon d'entrées/sorties** fourni à l'instruction **OUVRIR** est déjà associé à un autre **canal d'entrées/sorties**.

Il est interdit d'ouvrir un même document plus d'une fois simultanément, même sur des canaux différents. Pour corriger le problème, fermer premièrement le canal associé au document visé (avec l'instruction [FERMER](#)); ce dernier peut ensuite être réouvert sur un autre canal.

Consultez les sections portant sur les [canaux d'entrées/sorties](#) et l'[ouverture d'un canal d'entrées/sorties](#) pour plus d'information.

E2016 Impossible d'ouvrir le fichier ou tampon d'E/S spécifié

Il est impossible d'ouvrir le [fichier](#) ou [tampon d'entrées/sorties](#) spécifié dans l'instruction [OUVRIR](#).

Si le document à ouvrir est un fichier, l'erreur est probablement due à l'impossibilité d'accéder au support médiatique (disque, disquette ou autres), une défaillance de ce support médiatique ou à un fichier corrompu. Assurez-vous que le fichier existe et est accessible s'il est pour être lu, ou qu'il peut être supplanté s'il existe déjà mais doit être remplacé.

L'erreur peut aussi être causée par une erreur d'accès au [répertoire des fichiers temporaires](#). Ce répertoire est utilisé pour stocker des fichiers créés au besoin par *LARP* pour gérer la manipulation des tampons d'entrées/sorties.

Pour plus d'information sur les fichiers temporaires et la sélection du répertoire où sont stockés ces fichiers, consulter la section portant sur la [configuration de la console d'exécution](#).

E2017 Impossible d'ouvrir un fichier temporaire

Lors de l'[ouverture d'un tampon d'entrées/sorties](#), *LARP* crée un fichier temporaire afin de gérer les lectures et/ou écritures au tampon. Ce fichier temporaire est créé dans un répertoire indiqué par *Windows*[®]. Cette erreur indique que *Windows*[®] est dans l'impossibilité de fournir un tel répertoire à *LARP*.

Les causes probables de telles erreurs sont un manque d'espace sur le support médiatique (i.e. disque rigide), l'impossibilité d'accéder au support médiatique en écriture ou une défaillance de ce support médiatique.

L'erreur peut aussi être causée par une erreur d'accès au [répertoire des fichiers temporaires](#). Ce répertoire est utilisé pour stocker des fichiers créés par *LARP* pour gérer la manipulation des tampons d'entrées/sorties.

Pour plus d'information sur les fichiers temporaires et la sélection du répertoire où sont stockés ces fichiers, consulter la section portant sur la [configuration de la console d'exécution](#).

E2018 Impossible d'accéder au fichier ou tampon d'E/S spécifié

Un [fichier](#) ou un [tampon d'entrées/sorties](#) fut ouvert avec succès avec l'instruction [OUVRIR](#), mais une erreur s'est produite lors d'une [lecture](#) ou [écriture](#) via le canal d'entrées/sorties associé au document ouvert.

Les causes de telles erreurs peuvent être multiples, dont un manque d'espace sur le support médiatique (i.e. disque rigide), l'impossibilité soudaine d'accéder au support médiatique ou une défaillance de ce support médiatique.

E2019 Fin du fichier ou tampon d'E/S atteinte

Une tentative de [lecture](#) est effectuée dans un [fichier](#) ou un [tampon d'entrées/sorties](#) alors que la fin du document est atteinte (i.e. il n'y a plus de données à lire).

Exploitez la fonction prédéfinie [FINDECONTENU](#) pour détecter la fin d'un fichier ou tampon d'entrées/sorties en lecture.

E2020 Chaîne de formatage invalide

Une chaîne de formatage invalide est fournie à la fonction prédéfinie [FORMATER](#).

Consultez la documentation de cette fonction pour corriger le problème.

E2021 Débordement de capacité de la pile d'appels (récursivité infinie?)

L'exécution de l'algorithme cause une récursivité infinie. Lorsqu'un module fait appel à lui-même ou que deux modules s'invoquent mutuellement, il peut éventuellement y avoir épuisement de la mémoire de l'ordinateur si ce processus d'appel ne cesse pas. *LARP* a détecté une telle situation.

Consultez la section portant sur la [récursivité](#) pour plus d'information.

E2022 Valeur de variable ne pouvant pas être modifiée

Selon les circonstances, une [variable](#) peut parfois être verrouillée de sorte que sa valeur ne puisse être modifiée. Un exemple d'une telle situation est une variable d'itération ne pouvant être modifiée à l'intérieur d'une structure répétitive [POUR](#).

L'algorithme a tenté de modifier la valeur d'une variable verrouillée.

E2023 La variable d'itération «*nom de variable*» ne peut pas être modifiée dans la boucle

Une variable d'itération ne peut pas être modifiée par les instructions à l'intérieur d'une structure répétitive [POUR](#). Cette variable est implicitement et exclusivement modifiée par l'instruction [POUR](#) à chaque itération. L'algorithme a tenté de modifier explicitement une variable d'itération.

Consultez la section portant sur la structure répétitive [POUR](#) afin d'obtenir plus d'information.

E2024 La variable «*nom de variable*» contient un nom de fichier ou de tampon d'E/S invalide

Le nom fourni à l'instruction [OUVRIR](#) via la [variable](#) spécifiée est invalide.

Assurez-vous que la variable spécifiée contient le nom du [fichier](#) ou du [tampon d'entrées/sorties](#) à ouvrir. Si ce qui est spécifié n'est pas une variable mais le nom du document à ouvrir, vous avez probablement oublié de spécifier son nom entre guillemets tel une [chaîne de caractères](#).

Lors de l'[ouverture d'un fichier](#), le nom fourni est invalide s'il n'est pas conforme aux noms de fichier permis par *Windows*[®], ou si un fichier à ouvrir en lecture n'existe pas. Lors de l'[ouverture d'un tampon d'entrées/sorties](#), le nom de tampon fourni est invalide s'il correspond à aucun tampon existant du projet.

Consultez la section portant sur les [fichiers et tampons d'entrées/sorties](#) pour plus d'informations.

E2025 Nom de fichier ou de tampon d'E/S invalide (avez-vous oublié les guillemets?)

Le nom fourni à l'instruction **OUVRI**R est invalide. Vous avez peut-être oublié de spécifier son nom entre guillemets tel une [chaîne de caractères](#).

Lors de l'[ouverture d'un fichier](#), le nom fourni est invalide s'il n'est pas conforme aux noms de fichier permis par *Windows*[®], ou si un fichier à ouvrir en lecture n'existe pas. Lors de l'[ouverture d'un tampon d'entrées/sorties](#), le nom de tampon fourni est invalide s'il correspond à aucun tampon existant du projet.

Consultez la section portant sur les [fichiers et tampons d'entrées/sorties](#) pour plus d'information.

E2026 Boucle infinie causée par un incrément de signe opposé au sens de l'itération

Une [structure répétitive POUR](#) impose un incrément de variable d'itération contraire à l'orientation de la valeur de départ vers la valeur finale de cette variable. Un exemple d'une telle anomalie est une variable d'itération devant varier des valeurs 1 jusqu'à 10 alors que la valeur de la variable est réduite à chaque itération (**POUR** i = 1 **JUSQU'À** 10 **INCRÉMENT** -1 FAIRE).

Assurez-vous d'incrémenter la variable d'itération en fonction de la direction des valeurs limites de cette variable (dans l'exemple précédent, puisque i varie de 1 vers 10, la valeur de la variable doit augmenter à chaque itération).

Consultez la section portant sur la structure répétitive **POUR** afin d'obtenir plus d'information.

E2027 Nombre d'arguments en appel ne correspond pas au nombre de paramètres requis par le module

Un appel à un module auxiliaire défini dans le projet ne fournit pas le nombre d'arguments requis par ce module. Soit l'appel de module ne fournit pas assez d'arguments ou il en fournit trop.

Pour plus d'information sur la correspondance imposée aux arguments et aux paramètres de modules, consultez la section [Modules auxiliaires avec paramètres](#).

E2028 Un paramètre référence du module n'a pas une variable comme argument correspondant en appel

Un appel de module auxiliaire ne fournit une variable comme argument correspondant à un paramètre référence du module. Lorsqu'un module définit un paramètre référence, tout appel à ce module doit fournir comme argument correspondant le nom d'une variable pouvant recevoir la valeur déposée par le module dans son paramètre référence.

Pour plus d'information sur les paramètres références, consultez la section [Paramètres références](#).

E2029 Invocation d'un module non défini

L'algorithme fait appel à un module auxiliaire n'étant pas défini dans le projet.

Consultez la section sur les [modules auxiliaires](#) pour plus d'information.

E2101 Erreur «code d'erreur» dans le logiciel (adresse «adresse») - contactez le support technique

Ce message d'erreur indique qu'une erreur inconnue s'est produite lors de l'exécution de l'algorithme.

Contactez le [support technique](#) et communiquez l'information incluse dans le message d'erreur (le *code d'erreur* et l'*adresse* où l'erreur s'est produite dans *LARP*).

E2102 Erreur («description») dans le logiciel (adresse «adresse») - contactez le support technique

Ce message d'erreur indique qu'une erreur inconnue s'est produite lors de l'exécution de l'algorithme.

Contactez le [support technique](#) et communiquez l'information incluse dans le message d'erreur (la *description* de l'erreur et l'*adresse* en mémoire où l'erreur s'est produite dans *LARP*).

E2103 Le processeur a détecté une opération arithmétique invalide

Une erreur s'est produite au niveau des calculs arithmétiques dans le microprocesseur de l'ordinateur. Cette erreur est probablement due à un bogue dans *LARP*, telle qu'une opération arithmétique produisant un [résultat trop grand ou trop petit](#) pour être manipulé par l'ordinateur.

Cette erreur devrait rarement être affichée par *LARP*.

E2104 Valeur flottante trop grande pour être manipulée

Un calcul arithmétique dans l'algorithme a produit un résultat flottant trop grand pour être manipulé par l'ordinateur.

Pour plus d'information sur la plus grande valeur flottante permise, consultez la section portant sur les [numériques](#).

E2105 Valeur flottante trop petite pour être manipulée

Un calcul arithmétique dans l'algorithme a produit un résultat flottant trop petit pour être manipulé par l'ordinateur.

Pour plus d'information sur la plus petite valeur flottante permise, consultez la section portant sur les [numériques](#).

E2106 Tentative de division par zéro

Un calcul arithmétique dans l'algorithme comprend une division dont le dénominateur est 0.

E2107 Erreur mathématique inconnue impliquant une valeur flottante

Un calcul arithmétique impliquant un [flottant](#) comprend une opération valide du point de vue de la syntaxe, mais invalide lors de son évaluation.

En d'autres termes, le microprocesseur de l'ordinateur est incapable d'effectuer le calcul demandé.

E2108 Erreur mathématique inconnue impliquant une valeur entière

Un calcul arithmétique impliquant un [entier](#) comprend une opération valide du point de vue de la syntaxe, mais invalide lors de son évaluation.

En d'autres termes, le microprocesseur de l'ordinateur est incapable d'effectuer le calcul demandé.

E2109 Valeur entière trop grande ou index invalide

Un calcul arithmétique dans l'algorithme a produit un résultat entier trop grand pour être manipulé par l'ordinateur.

Pour plus d'information sur la plus grande valeur entière permise, consultez la section portant sur les [numériques](#).

L'erreur peut aussi occasionnellement être causée par un [accès à un élément de conteneur](#) inexistant.

E2110 Valeur entière trop grande pour être manipulée

Un calcul arithmétique dans l'algorithme a produit un résultat entier trop grand pour être manipulé par l'ordinateur.

Pour plus d'information sur la plus grande valeur entière permise, consultez la section portant sur les [numériques](#).

E2111 Capacité de la mémoire vive atteinte

Il n'y a plus de mémoire vive disponible dans l'ordinateur pour poursuivre l'exécution de l'algorithme. La cause peut être un trop grand nombre de [conteneurs](#) surdimensionnés, ou un trop grand nombre d'applications autres que LARP s'exécutant simultanément.

Pour corriger le problème fermez toutes les applications non essentielles. Si vous exploitez des conteneurs, réduisez leurs dimensions.

E2112 Fichier introuvable

Le nom de fichier fourni à l'instruction [OUVRIR](#) en [mode LECTURE](#) est invalide. Il est fort probable que le fichier n'existe pas ou, s'il existe, qu'il soit inaccessible (peut être verrouillé par une autre application).

E2113 Nom de fichier invalide

Le nom de fichier fourni à l'instruction [OUVRIR](#) est invalide. Le nom du fichier doit être conforme aux exigences du système de fichiers de *Windows*[®].

Consultez la section portant sur les [fichiers et tampons d'entrées/sorties](#) pour plus d'information.

E2114 Trop de fichiers ouverts simultanément

Le système d'exploitation *Windows*[®] permet un nombre limité de fichiers pouvant être ouverts simultanément. Une tentative d'ouverture de fichier (effectuée explicitement dans l'algorithme via l'instruction [OUVRIR](#), ou implicitement à l'ouverture d'un [fichier temporaire](#)) a échoué car ce maximum est atteint.

Pour corriger le problème, consultez la documentation de *Windows*[®] afin de déterminer le maximum de fichiers ouverts permis et modifiez votre algorithme de façon à ne pas ouvrir plus de [fichiers](#) ou [tampons d'entrées/sorties](#) que permis.

E2115 Accès refusé au fichier

Le nom de [fichier](#) fourni à l'instruction **OUVRIR** correspond à un fichier auquel *LARP* n'a pas accès.

L'erreur peut être due à une tentative d'ouvrir un fichier en [mode ÉCRITURE](#) ou [AJOUT](#) alors qu'aucune modification n'est permise à ce fichier. Il se peut aussi que le fichier soit temporairement verrouillé par une autre application.

Pour corriger le problème, assurez-vous que le fichier visé existe et est accessible selon le mode d'accès désiré.

L'erreur peut aussi être causée par un accès invalide au [répertoire des fichiers temporaires](#). Ce répertoire est utilisé pour stocker des fichiers créés par *LARP* lors de la manipulation de [tampons d'entrées/sorties](#).

Pour plus d'information sur les fichiers temporaires et la sélection du répertoire où sont stockés ces fichiers, consultez la section portant sur la [configuration de l'exécution d'algorithmes](#).

E2116 Fin du fichier atteinte

Une tentative de [lecture](#) est effectuée dans un [fichier](#) alors que la fin du fichier est atteinte (i.e. il n'y a plus de données à lire).

Exploitez la fonction prédéfinie **FINDECONTENU** pour détecter la fin d'un fichier ouvert en [mode LECTURE](#).

E2117 Capacité du support médiatique atteinte

L'espace disponible sur le support médiatique (disque, disquette ou autres) où est stocké le [fichier](#) manipulé par l'algorithme est épuisée. L'erreur peut aussi être causée par la manipulation de [tampons d'entrées/sorties](#) puisque ceux-ci sont associés à des [fichiers temporaires](#) nécessitant de l'espace disque.

Pour corriger le problème, libérez de l'espace disque sur le support médiatique ou utilisez un support médiatique plus volumineux.

E2118 Données lues invalides

Une instruction **LIRE** visant un [canal d'entrées/sorties](#) associé à un [fichier](#) ouvert en [mode LECTURE](#) a échoué. La cause probable est une donnée de format invalide retrouvée dans le fichier en question. Il se peut aussi que le fichier ou son contenu soit corrompu.

Pour plus d'information, consultez les sections portant sur les [fichiers et tampons d'entrées/sorties](#), ainsi que sur la [lecture via un canal d'entrées/sorties](#).

E2119 Erreur de fichier inconnue

Une erreur de nature inconnue s'est produite lors de la manipulation d'un [fichier](#) (à l'[ouverture](#), à la [lecture](#), à l'[écriture](#) ou à la [fermeture](#)). Il se peut aussi que le fichier ou son contenu soit corrompu.

Consultez la section portant sur les [fichiers et tampons d'entrées/sorties](#) pour plus d'information.

E2120 Capacité de la pile d'exécution atteinte (peut être dû à une récursivité infinie)

L'exécution de l'algorithme cause une récursivité infinie. Lorsqu'une fonction s'invoque elle-même ou lorsque deux fonctions s'invoquent mutuellement, il peut éventuellement y avoir épuisement de la mémoire de l'ordinateur si ce processus d'appels ne cesse pas. *LARP* a détecté une telle situation.

Consultez la section portant sur la [récursivité](#) pour plus d'information. La taille de la pile d'exécution peut aussi être augmentée (consultez la section portant sur la [configuration de l'exécution d'algorithmes](#)).

Si l'algorithme n'est pas en cause, contactez le [support technique](#) afin de les informer du problème.

E2121 Opération impliquant des types de données incompatibles

Certains types de [variables](#) ou de [valeurs numériques](#) ne peuvent pas être combinés dans une expression arithmétique. Un exemple évident est de tenter de diviser un nombre par une [chaîne de caractères](#) (ex : **10/'allo'**).

Lors de l'exécution de telles expressions, *LARP* interrompt l'exécution de l'algorithme et affiche ce message d'erreur.

E2122 Erreur d'accès à un fichier («*description*»)

Ce message d'erreur indique qu'une erreur s'est produite lors d'une tentative d'accès à un [fichier](#) ou un [tampon d'entrées/sorties](#). Un message d'erreur retourné par l'ordinateur décrit brièvement le type d'erreur rencontrée.

La cause la plus probable de cette erreur est l'impossibilité d'accéder au fichier spécifié car celui-ci est subitement devenu inaccessible. Assurez-vous que le support médiatique (disque, disquette, etc.) contenant le fichier spécifié est opérationnel et accessible.

L'erreur peut aussi être causée lors d'un accès invalide au [répertoire des fichiers temporaires](#). Ce répertoire est utilisé pour stocker des fichiers créés par *LARP* afin de gérer les accès aux tampons d'entrées/sorties.

Pour plus d'information sur les fichiers temporaires et la sélection du répertoire où sont stockés ces fichiers, consultez la section portant sur la [configuration de l'exécution d'algorithmes](#).

E2999 Erreur inconnue; contactez le support technique

Une erreur non prévue par les développeurs de *LARP* fut détectée. Veuillez contacter le [support technique](#) de *LARP* afin d'obtenir de l'aide.

E3001 Des tampons d'E/S ou des fichiers n'ont pas été fermés

L'algorithme exécuté a ouvert des fichiers ou tampons d'entrées/sorties (avec l'instruction [OUVRIR](#)) mais n'a pas fermé ceux-ci avant de terminer son exécution.

Consultez l'aide en ligne portant sur l'instruction [FERMER](#) pour plus d'information.

E3002 Variable «nom de variable» sans valeur

L'algorithme fait référence à la valeur d'une **variable** alors qu'aucune valeur n'a précédemment été attribuée à cette variable.

Assurez-vous d'affecter une valeur à la variable avant de l'employer dans des calculs ou d'autres instructions *LARP*. Il est aussi possible que vous aillez fait une erreur d'épellation dans le nom de la variable, ce qui la rend distinct d'une autre variable à laquelle une valeur fut attribuée.

Lorsqu'un algorithme fait référence à une variable n'ayant pas de valeur attribuée, on dit cette variable *indéterminée*. Les variables indéterminées causent parfois des erreurs fatales lors de l'exécution d'algorithmes.

Pour plus d'information, consultez la section portant sur [l'affectation](#).

E3003 Variable «nom de variable» sans valeur (confondue avec la variable «nom de variable»?)

L'algorithme fait référence à la valeur d'une **variable** alors qu'aucune valeur n'a précédemment été attribuée à cette variable.

Assurez-vous d'affecter une valeur à la variable avant de l'employer dans des calculs ou d'autres instructions *LARP*. Il est aussi possible que vous aillez fait une erreur d'épellation dans le nom de la variable, ce qui la rend distinct d'une autre variable à laquelle une valeur fut attribuée.

Lorsqu'un algorithme fait référence à une variable n'ayant pas de valeur attribuée, on dit cette variable *indéterminée*. Les variables indéterminées causent parfois des erreurs fatales lors de l'exécution d'algorithmes.

Pour plus d'information, consultez la section portant sur [l'affectation](#).

E3004 Élément du conteneur «nom de conteneur» sans valeur

L'algorithme fait référence à la valeur d'un élément inexistant de **conteneur**.

Assurez-vous d'affecter une valeur à la position de conteneur référée avant de l'employer dans des calculs ou dans d'autres instructions *LARP*. Il est aussi possible que vous aillez fait une erreur d'épellation dans le nom du conteneur, ce qui le rend distinct d'un autre conteneur auquel des valeurs furent attribuées. Enfin, assurez-vous de l'index de l'élément référé est valide.

Lorsqu'un algorithme fait référence à un élément de conteneur n'ayant pas de valeur attribuée, on dit cet élément *indéterminé*. Les variables et éléments de conteneur indéterminés causent parfois des erreurs fatales lors de l'exécution d'algorithmes.

Pour plus d'information, consultez la section portant sur [l'affectation](#).

E3005 Élément du conteneur «nom de conteneur» sans valeur (confondue avec la variable «nom de variable»?)

L'algorithme fait référence à la valeur d'un élément inexistant de **conteneur**.

Assurez-vous d'affecter une valeur à la position de conteneur référée avant de l'employer dans des calculs ou dans d'autres instructions *LARP*. Il est aussi possible que vous aillez fait une erreur d'épellation dans le nom du conteneur, ce qui le rend distinct d'un autre conteneur

auquel des valeurs furent attribuées; *LARP* a identifié un autre conteneur ou [variable](#) ayant une valeur attribuée et dont le nom est semblable à celle erronée. Enfin, assurez-vous de l'index de l'élément référé est valide.

Lorsqu'un algorithme fait référence à un élément de conteneur n'ayant pas de valeur attribuée, on dit cet élément *indéterminé*. Les variables et éléments de conteneur indéterminés causent parfois des erreurs fatales lors de l'exécution d'algorithmes.

Pour plus d'informations, consultez la section portant sur [l'affectation](#).

E3006 Appel d'un module ne retournant pas de valeur

Un module ne retournant pas de valeur de retour est invoqué comme si une valeur était retournée.

Pour plus d'information, consultez la section portant sur les [modules avec valeur de retour](#).

E3007 Accès à un élément de conteneur sans valeur

L'algorithme fait référence à la valeur d'un élément inexistant de [conteneur](#).

Assurez-vous d'affecter une valeur à la position de conteneur référée avant de l'employer dans des calculs ou dans d'autres instructions *LARP*. Il est aussi possible que vous aillez fait une erreur d'épellation dans le nom du conteneur, ce qui le rend distinct d'un autre conteneur auquel des valeurs furent attribuées. Enfin, assurez-vous de l'index de l'élément référé est valide.

Lorsqu'un algorithme fait référence à un élément de conteneur n'ayant pas de valeur attribuée, on dit cet élément *indéterminé*. Les variables et éléments de conteneur indéterminés causent parfois des erreurs fatales lors de l'exécution d'algorithmes.

E3008 Les boucles POUR des organigrammes de ce projet doivent être vérifiées car elles peuvent ne pas correspondre au nouveau format de structures répétitives inconditionnelles

Le format d'une boucle [POUR](#) sous forme d'organigramme a dû être modifié afin de rendre sa syntaxe plus conforme à la version pseudo-code de la structure [POUR](#).

Vous avez ouvert un fichier projet qui fut créé avec une version antérieure de *LARP*, et la version actuelle a dû convertir au nouveau format les structures [POUR](#) contenus dans les organigrammes du fichier projet. Puisque la conversion ne peut garantir l'intégrité de la structure, il est conseillé de réviser les structures [POUR](#) des organigrammes du projet afin d'en valider les composants (la variable d'itération, les valeurs de départ et de fin, et l'incrément).

Pour de l'information sur la structure répétitive [POUR](#) et sa syntaxe, consultez la section [Structure POUR](#).

E9999 Erreur inconnue; contactez le support technique

Une erreur non prévue par les développeurs de *LARP* fut détectée. Veuillez contacter le [support technique](#) de *LARP* afin d'obtenir de l'aide.

Index

-	Opérateurs arithmétiques	113
	Opérateurs de conteneurs	115
	Priorité des opérateurs	125
'	(chaîne de caractères)	<i>Voir Apostrophe</i>
\$	(continuation d'instruction)	91
	Structure SI-SINON-SI	133
//	Les commentaires	89
	Ouvrir un fichier	162
	Séquences d'échappement	100
=	Affectation	98
"	(chaîne de caractères)	<i>Voir Guillemet</i>
\	(séquence d'échappement)	<i>Voir Séquence d'échappement</i>
=	(affectation)	<i>Voir Affectation</i>
#	IND	<i>Voir Indéfini</i>
[]	(conteneur)	<i>Voir Élément de conteneur</i>
,	(virgule)	
	Déclaration des paramètres d'un module	150
	L'instruction de lecture	106
	L'instruction d'écriture	107
	L'instruction de requête	109
	Regroupement de valeurs	101
+	Opérateurs arithmétiques	113
	Priorité des opérateurs	125
*	Opérateurs arithmétiques	113
	Priorité des opérateurs	125
/	Opérateurs arithmétiques	113
	Priorité des opérateurs	125
//	Opérateurs arithmétiques	113
	Priorité des opérateurs	125
%	Opérateurs arithmétiques	113
	Priorité des opérateurs	125
^	Opérateurs arithmétiques	113
	Priorité des opérateurs	125
+	Opérateurs de chaînes de caractères	114
	Opérateurs de conteneurs	115
<	Opérateurs relationnels	122
	Priorité des opérateurs	125
<=	Opérateurs relationnels	122
	Priorité des opérateurs	125
>	Opérateurs relationnels	122
	Priorité des opérateurs	125
>=	Opérateurs relationnels	122
	Priorité des opérateurs	125
=	Opérateurs relationnels	122
!=	Opérateurs relationnels	122
	Priorité des opérateurs	125
<>	(inégalité)	<i>Voir !=</i>
!	Opérateurs logiques	125
	Priorité des opérateurs	125
=	Priorité des opérateurs	125
()	Déclaration des paramètres d'un module	151
	Opérateurs arithmétiques	113
	Priorité des opérateurs	125
:	(double point)	136
A		
ABS	<i>Voir ABSOLU</i>
ABSOLU	116, 175
Accents	Noms de variables	97
	Syntaxe des instructions	91
Achat	Commander des clés de débrillage	26
	Procédure d'enregistrement	24
Addition	<i>Voir +</i>
Affectation	100
	Accès aux éléments	102
	Animation	68
	L'instruction de lecture	106
	Modules avec valeur de retour	157
	Noms de variables	98
	Opérations	98
Agrandissement	57
	Commandes de l'éditeur graphique accessibles via le clavier	59
	Panneau de statut	45
Aide abrégée	Aide disponible dans LARP	33
	Éléments de l'interface	35
	Panneau de statut	45
	Support technique	27
Aide en ligne	27
	Aide disponible dans LARP	33
	Avertissements et erreurs	71
	Introduction	18
	Support technique	27
	Syntaxe des instructions	91
AJOUT	163
	Écriture via un canal d'entrées/sorties	166
	Syntaxe (fichier)	205
	Syntaxe (tampon d'entrées/sorties)	204
ALÉA	<i>Voir ALÉATOIRE</i>
ALÉATOIRE	116, 176
Algorithme	Introduction	17
	Un premier algorithme	89
ALORS	Structure SI-SINON-SI	130
	Structures conditionnelles	121
	Structures SI et SI-SINON	126
	Syntaxe (structure SI)	201
	Syntaxe (structure SI-SINON)	201
	Syntaxe (structure SI-SINON-SI)	202
Animation	68
	Interface de l'exécution pas-à-pas	63
	Modes d'exécution pas-à-pas	64
	Opérateurs arithmétiques	113
	Priorité des opérateurs	126

Annulation d'opérations	72	Canal d'entrées/sorties.....	161
Apostrophe	99	Détection de fin de contenu via un canal	
ARCTAN.....	<i>Voir ARCTANGENTE</i>	d'entrées/sorties	166
ARCTANGENTE.....	116, 177	Écriture via un canal d'entrées/sorties.....	165
Argument		Fermeture d'un canal d'entrées/sorties	164
Déclaration des paramètres d'un module.....	151	Fichiers et tampons d'entrées/sorties.....	159
Paramètres références	153	FINDECONTENU	184
Paramètres valeurs	152	Lecture via un canal d'entrées/sorties	164
Syntaxe des instructions.....	90	Modes d'accès.....	163
ARR.....	<i>Voir ARRONDIR</i>	Ouverture d'un document.....	161
ARRONDIR	116, 178	CAP	<i>Voir CAPACITÉ</i>
ASCII		CAPACITÉ.....	118, 179
La numérotation en base binaire	168	COMPTER.....	180
Opérateurs relationnels	123	Retirer des éléments.....	103
Auteur de LARP.....	21	Caractères accentués	<i>Voir Accents</i>
Avertissements	70, 207	CD	
Aide en ligne	27	Exigences minimales d'équipements et logiciels ...	21
Compilation et exécution	60	Installation à partir d'un CD.....	22
		CHAÎNE	
		Tests de type	123
		Chaîne de caractères	99
		COMPTER.....	179
		Couleurs dans les éditeurs.....	77
		ENCARACTÈRES	182
		ENCHAÎNE.....	183
		Fonctions de manipulation de chaînes	117
		FORMATER	185
		L'instruction de lecture	106
		MAJUSCULES	191
		MINUSCULES	193
		Opérateurs de chaînes de caractères	114
		Opérateurs relationnels.....	123
		POSITION	196
		Sélection de couleurs.....	76
		Séquences d'échappement.....	100
		SOUSENSEMBLE	198
		Chaîne de formatage	
		FORMATER	185
		Chiffrement	
		Chiffrement des documents	81
		Contrôle du copier-coller.....	82
		Clavier	
		Commandes de l'éditeur textuel accessibles via le	
		clavier	50
		Commandes d'édition de l'éditeur graphique.....	58
		Commandes d'édition de l'éditeur textuel.....	49
		Détection de fin de contenu via un canal	
		d'entrées/sorties	166
		Fichiers et tampons d'entrées/sorties.....	159
		L'instruction de lecture	106
		Clé de débridage	
		Commander des clés de débridage	25
		Configuration du mode super-utilisateur et du	
		système de mises à jour.....	74
		Mode super-utilisateur	79
		Port parallèle	84
		Port USB	84
		Sélection d'une technologie de clé.....	84
		Clé d'enregistrement	24
		Codage binaire.....	167
		Codage décimal	167
		Codage hexadécimal	170
		Coller	
		Insertion, déplacement et destruction d'instructions	
		d'organigramme	55
		Commandes	
		Barre de menu.....	35
		Commandes de l'éditeur graphique accessibles via	
		le clavier.....	59

B**C**

DÉFINIE		Panneau de messages	44
Tests de type	124	Points d'arrêt	67
Démarrage de LARP	21	Surligne de contenu	49
Configuration du mode super-utilisateur et du		Éditeurs	42
système de mises à jour	75	Configuration des éditeurs	72
Mises à jour de LARP	26	Éléments de l'interface	34
Déplacement		Panneau de messages	44
Insertion, déplacement et destruction d'instructions		Élément de conteneur	101
d'organigramme	55	Accès aux éléments	101
Désinstallation	23	Retirer des éléments	102
Enregistrement	23	Éléments de l'interface	33
Licence d'utilisation	19	EN	
DÉTERMINÉE		Syntaxe (fichier)	205
Tests de type	124	Syntaxe (tampon d'entrées/sorties)	204
DÉTRUIRE	102	ENCARAC	<i>Voir ENCARACTÈRES</i>
Différence		ENCARACTÈRES	117, 118, 182
Opérateurs de conteneurs	115	ENCHAÎNE	117, 118, 183
Distribution	20	FORMATER	185
Distribution de LARP		Enregistrement	23
Licence d'utilisation	19	Licence d'utilisation	18
Division entière	113, <i>Voir //</i>	Mises à jour de LARP	26
Division flottante	<i>Voir /</i>	Procédure d'enregistrement	24
Document		Enseignement	
Éditeurs	42	Chiffrement des documents	82
Navigateur de documents	41	Conversion d'organigramme en pseudo-code	86
Ouverture d'un document	161	Débridage de l'environnement de développement	83
Documentation	18	Introduction	17
Droits d'auteur	21	Mode super-utilisateur	79
		Projets publics	86
		Statistiques de projet	85
		Entier	
		<i>L'instruction de lecture</i>	106
		Sélection de couleurs	76
		Valeurs numériques	99
		ENTIER	
		Tests de type	123
		Entrées/sorties	105
		Instruction d'entrées/sorties pour organigrammes	105
		ENTRER	201
		Déclaration des paramètres d'un module	150
		Éditer un organigramme	53
		Modules auxiliaires	147
		Environnement de développement	33
		Aide en ligne	27
		Configuration générale	71
		Éléments de l'interface	33
		Introduction	17
		Messages reliés à l'environnement de	
		développement	207
		Sélection de couleurs	76
		Tampons d'entrées/sorties	159
		Erreurs	
		Aide disponible dans LARP	33
		Aide en ligne	27
		Avertissements et erreurs	70
		Avertissements et erreurs	207
		Compilation et exécution	60
		Erreur d'application	28
		Panneau de messages	44
		Rapporter un bogue	28
		Espace	
		Le séparateur	109
		EST	123
		ET	
		Opérateurs logiques	125
		Priorité des opérateurs	125
		EXÉCUTER	
		Modules auxiliaires	147
		Modules auxiliaires avec paramètres	150
		Syntaxe d'invocation alternative	157

Exécution.....	60	FIN.....	201
Avertissements et erreurs.....	70	Début et fin d'un algorithme	90
Configuration de la console d'exécution.....	73	Éditer un organigramme	53
Console d'exécution	45	Module principal.....	146
Début et fin d'un algorithme.....	90	FINDECONTENU.....	166, 184
Exécution d'un projet.....	61	FINPOUR.....	204
Instructions d'organigramme	53	Structure POUR.....	141
Messages reliés à l'exécution d'algorithmes	210	FINSÉLECTIONNER	202
Panneau de messages.....	44	Structure de sélection	134
Sauvegarde de sécurité.....	70	FINSI	
Valeurs numériques	99	Structure SI-SINON-SI.....	130
Exécution pas-à-pas	61	Structures conditionnelles.....	121
Animation.....	68	Structures SI et SI-SINON	126
Couleurs pour exécution pas-à-pas	77	Syntaxe (structure SI).....	201
Éditer un document textuel.....	47	Syntaxe (structure SI-SINON).....	201
Fenêtre d'exécution pas-à-pas	46	Syntaxe (structure SI-SINON-SI)	202
Inspection de la pile d'appels.....	66	FINTANTQUE	203
Inspection des variables.....	65	Structure TANTQUE	137
Interface de l'exécution pas-à-pas	61	Flottant	
Modes d'exécution pas-à-pas.....	64	L'instruction de lecture.....	106
Opérateurs arithmétiques	113	Sélection de couleurs.....	76
Points d'arrêt.....	66	Valeurs numériques.....	99
Priorité des opérateurs	126	FLOTTANT	
Sélection de couleurs	76	Tests de type	123
Surligne d'instructions en exécution pas-à-pas.....	57	Fonction prédéfinie	
Surligne de contenu	49	Fonctions de manipulation de chaînes.....	117
Exigences logicielles.....	22	Fonctions de manipulation de conteneurs.....	118
Exigences matérielles.....	21	Fonctions mathématiques.....	116
EXP	117, 183	Fonctions trigonométriques.....	<i>Voir Fonctions mathématiques</i>
LOGE.....	190	Fonctions de canal d'entrées/sorties	
Exposant	<i>Voir ^</i>	FINDECONTENU	184
		Fonctions de chaîne	
		COMPTER.....	179
		FORMATER	185
		MAJUSCULES	191
		MINUSCULES.....	193
		POSITION	196
		SOUSENSEMBLE	198
		Fonctions de conteneur	
		CAPACITÉ.....	179
		COMPTER.....	179
		DATE.....	181
		ENCARACTÈRES	182
		ENCHAÎNE.....	183
		HEURE.....	189
		MAXIMUM	191
		MINIMUM	192
		POSITION	196
		SOUSENSEMBLE	198
		Fonctions mathématiques	<i>Voir Fonctions trigonométriques</i>
		ABSOLU.....	175
		ALÉATOIRE	176
		ARRONDIR	178
		EXP	183
		LOG10.....	189
		LOGE	190
		MAXIMUM	191
		MINIMUM	192
		PLAFOND	194
		PLANCHER.....	195
		RACINE.....	197
		Fonctions prédéfinies	116, 175
		ABSOLU.....	175
		ALÉATOIRE	176
		ARCTANGENTE.....	177
		ARRONDIR	178
		CAPACITÉ.....	179
		COMPTER.....	179

F

F1 (aide).....	33
Facteur d'agrandissement	
Agrandissement de l'affichage.....	57
Factorielle (récursivité).....	173
FAIRE	
Structure POUR	141
Structure TANTQUE.....	137
Syntaxe (structure POUR).....	204
Syntaxe (structure TANTQUE)	203
FDC.....	<i>Voir FINDECONTENU</i>
Fenêtre d'accueil	92
Commander des clés de débridage.....	25
Fenêtre d'enregistrement.....	92
Procédure d'enregistrement	24
FERMER	164
Syntaxe.....	205
Fichier.....	160
Canaux d'entrées/sorties.....	161
Détection de fin de contenu via un canal d'entrées/sorties.....	166
Écriture via un canal d'entrées/sorties	165
Fermeture d'un canal d'entrées/sorties.....	164
Fichiers et tampons d'entrées/sorties	159
FINDECONTENU.....	184
Lecture et écriture	105
Lecture via un canal d'entrées/sorties.....	164
Modes d'accès	163
Ouvrir un fichier.....	162
FICHER	205
Ouvrir un fichier	162
Fichier projet	
Chiffrement des documents.....	82
Rapporter un bogue	29
Fichiers temporaires	
Configuration de la console d'exécution.....	74

COSINUS.....	180
Couleurs dans les éditeurs.....	77
DATE.....	181
ENCARACTÈRES.....	182
ENCHAÎNE.....	183
EXP.....	183
FINDECONTENU.....	184
FORMATER.....	185
HEURE.....	189
LOG10.....	189
LOGE.....	190
MAJUSCULES.....	191
MAXIMUM.....	191
MINIMUM.....	192
MINUSCULES.....	193
PI (π).....	194
PLAFOND.....	194
PLANCHER.....	195
POSITION.....	196
RACINE.....	197
Retirer des éléments.....	103
SINUS.....	197
SOUSENSEMBLE.....	198
Syntaxe d'invocation alternative.....	158
Fonctions trigonométriques.....	<i>Voir Fonctions mathématiques</i>
ARCTANGENTE.....	177
COSINUS.....	180
PI (π).....	194
SINUS.....	197
FORMATER.....	118, 185
ENCHAÎNE.....	183
Freeware.....	<i>Voir Gratuitiel</i>
G	
Gestionnaire de clés.....	75
Glisser-déposer.....	
Contrôle de l'éditeur graphique via la souris.....	60
Contrôle de l'éditeur textuel via la souris.....	51
Éditer un organigramme.....	54
Fonctionnalités de l'éditeur graphique.....	51
Insertion, déplacement et destruction d'instructions d'organigramme.....	55
Panneau de modèles.....	42
Glisser-déposer:.....	58
Gratuitiel.....	
Conversion d'organigramme en pseudo-code.....	86
Création du projet <i>LARP</i>	92
Installation.....	21
Licence d'utilisation.....	18
Projets publics.....	87
Statistiques de projet.....	85
Guide d'utilisation.....	33
Guillemet.....	99
H	
Hasard.....	<i>Voir ALÉATOIRE</i>
HEURE.....	189
Hexadécimal.....	<i>Voir Codage hexadécimal</i>
FORMATER.....	187
I	
Impression.....	
Contrôle de l'impression.....	83
Éditer un document textuel.....	48
INCRÉMENT.....	
Structure POUR.....	141
Syntaxe (structure POUR).....	204
Indéfini.....	
L'affectation.....	100
Opérateurs arithmétiques.....	113
Opérateurs de conteneurs.....	115
Retirer des éléments.....	102
INDÉFINIE.....	
Tests de type.....	124
Indentation.....	<i>Voir Tabulation</i>
INDÉTERMINÉE.....	
Tests de type.....	124
Insertion.....	
Insertion, déplacement et destruction d'instructions d'organigramme.....	55
Inspection.....	
Inspection de la pile d'appels.....	66
Interface de l'exécution pas-à-pas.....	63
Installation.....	
Exigences minimales d'équipements et logiciels.....	21
Licence d'utilisation.....	18
Mises à jour de <i>LARP</i>	26
Instruction.....	
Séparation des instructions.....	91
Syntaxe des instructions.....	90
Instruction conditionnelle.....	
Structures conditionnelles.....	121
Instruction d'organigramme.....	
Structure SI-SINON-SI.....	133, 136
Instruction d'organigramme.....	52
Manipulation d'instructions d'organigramme.....	54
Instruction séquentielle.....	
Le séparateur.....	111
Opérations.....	98
Interface.....	
Fichiers et tampons d'entrées/sorties.....	159
Internet.....	
Auteur de <i>LARP</i>	21
Commander des clés de débridage.....	26
Mises à jour de <i>LARP</i>	27
Procédure d'enregistrement.....	24
Interruption d'exécution.....	
Console d'exécution.....	45
Exécution d'un projet.....	61
Introduction.....	17
Invocation.....	<i>Voir EXÉCUTER</i>
Configuration de la console d'exécution.....	73
Déclaration des paramètres d'un module.....	151
Itération.....	138
Structure RÉPÉTER-JUSQU'À.....	139
J	
Java.....	
Constantes et variables.....	97
Conteneurs.....	101
Introduction.....	17
L'affectation.....	100
Opérateurs et fonctions prédéfinies.....	113
Structure POUR.....	144
Journalisation.....	29
JUSQU'À.....	
Structure RÉPÉTER-JUSQU'À.....	139
JUSQU'À.....	
Structure POUR.....	141
JUSQU'À.....	
Syntaxe (structure RÉPÉTER-JUSQU'À).....	203
JUSQU'À.....	
Syntaxe (structure POUR).....	204

L		
Langage de programmation		E1007..... 211
Introduction.....	17	E1008..... 212
Syntaxe des instructions.....	91	E1009..... 212
LARP		E1010..... 212
Introduction.....	17	E1011..... 212
Lecture.....	106	E1012..... 212
Chaîne contenant des espaces.....	110	E1999..... 212
FINDECONTENU.....	184	E2001..... 212
Le séparateur.....	109	E2002..... 213
L'instruction de requête.....	108	E2003..... 213
LECTURE.....	163	E2004..... 213
Lecture via un canal d'entrées/sorties.....	165	E2005..... 213
Syntaxe (fichier).....	205	E2006..... 213
Syntaxe (tampon d'entrées/sorties).....	204	E2007..... 213
Licence.....	19	E2008..... 213
Licence d'utilisation.....	18	E2009..... 213
Enregistrement.....	23	E2010..... 214
LIRE.....	106, 205	E2011..... 214
Console d'exécution.....	45	E2012..... 214
Le séparateur.....	109	E2013..... 214
Lecture via un canal d'entrées/sorties.....	164	E2014..... 214
Modes d'accès.....	163	E2015..... 214
Lisp		E2016..... 215
Constantes et variables.....	97	E2017..... 215
LOG10.....	117, 189	E2018..... 215
LOGE.....	117, 190	E2019..... 216
EXP.....	183	E2020..... 216
LONGUEUR.....	118, Voir <i>COMPTER</i>	E2021..... 216
		E2022..... 216
		E2023..... 216
		E2024..... 216
		E2025..... 217
		E2026..... 217
		E2027..... 217
		E2028..... 217
		E2029..... 217
		E2101..... 218
		E2102..... 218
		E2103..... 218
		E2104..... 218
		E2105..... 218
		E2106..... 218
		E2107..... 218
		E2108..... 219
		E2109..... 219
		E2110..... 219
		E2111..... 219
		E2112..... 219
		E2113..... 219
		E2114..... 219
		E2115..... 220
		E2116..... 220
		E2117..... 220
		E2118..... 220
		E2119..... 220
		E2120..... 221
		E2121..... 221
		E2122..... 221
		E2999..... 221
		E3001..... 221
		E3002..... 222
		E3003..... 222
		E3004..... 222
		E3005..... 222
		E3006..... 223
		E3007..... 223
		E3008..... 223
		E9999..... 223
		MIN..... Voir <i>MINIMUM</i>
		MINIMUM..... 117, 118, 192
M		
Majuscules		
Noms de variables.....	97	
MAJUSCULES.....	118, 191	
Majuscules et minuscules		
Conversion.....	191, 193	
Syntaxe des instructions.....	91	
Marcher		
Modes d'exécution pas-à-pas.....	64	
Marge de numérotation		
Configuration des éditeurs.....	73	
Couleurs dans les éditeurs.....	77	
MAX.....	Voir <i>MAXIMUM</i>	
MAXIMUM.....	117, 118, 191	
Menu contextuel		
Avertissements et erreurs.....	71	
Commandes de l'éditeur graphique accessibles via les menus.....	58	
Commandes de l'éditeur textuel accessibles via les menus.....	50	
Commandes d'édition de l'éditeur graphique.....	58	
Commandes d'édition de l'éditeur textuel.....	49	
Contrôle de l'éditeur graphique via la souris.....	60	
Contrôle de l'éditeur textuel via la souris.....	51	
Éditer un document textuel.....	48	
Insertion, déplacement et destruction d'instructions d'organigramme.....	55	
Navigateur de documents.....	41	
Panneau de messages.....	44	
Messages		
Avertissements et erreurs.....	70	
Messages reliés à l'exécution d'algorithmes		
E1001.....	210	
E1002.....	211	
E1003.....	211	
E1004.....	211	
E1005.....	211	
E1006.....	211	

P	
Panneau de contrôle.....	39
Aide disponible dans <i>LARP</i>	33
Barre de menu.....	35
Commandes de l'éditeur graphique accessibles via les menus.....	59
Commandes de l'éditeur textuel accessibles via les menus.....	50
Commandes d'édition de l'éditeur graphique.....	58
Commandes d'édition de l'éditeur textuel.....	49
Éditer un document textuel.....	47
Éditer un organigramme.....	53
Éléments de l'interface.....	34
Interface de l'exécution pas-à-pas.....	62
Panneau de messages.....	44
Aide disponible dans <i>LARP</i>	33
Avertissements et erreurs.....	70
Compilation et exécution.....	60
Éléments de l'interface.....	34
Messages reliés à l'exécution d'algorithmes.....	210
Panneau de modèles.....	41
Commandes d'édition de l'éditeur graphique.....	58
Contrôle de l'éditeur graphique via la souris.....	60
Contrôle de l'éditeur textuel via la souris.....	51
Éléments de l'interface.....	34
Fonctionnalités de l'éditeur graphique.....	51
Insertion, déplacement et destruction d'instructions d'organigramme.....	55
Instructions d'organigramme.....	52
Panneau de statut.....	44
Aide disponible dans <i>LARP</i>	33
Débridage de l'environnement de développement.....	84
Éléments de l'interface.....	35
Inspection des variables.....	65
Interface de l'exécution pas-à-pas.....	63, 66
Paramètre.....	150
Déclaration des paramètres d'un module.....	150
Paramètres références.....	153
Paramètres valeurs.....	152
Paramètre référence.....	153
Déclaration des paramètres d'un module.....	152
Modules auxiliaires avec paramètres.....	150
Modules avec valeur de retour.....	157
Paramètre valeur.....	152
Déclaration des paramètres d'un module.....	152
Modules auxiliaires avec paramètres.....	150
Parenthèses.....	<i>Voir ()</i>
Partagiciel	
Chiffrement des documents.....	81, 82
Contrôle de l'impression.....	83
Conversion d'organigramme en pseudo-code.....	86
Création du projet <i>LARP</i>	92
Débridage de l'environnement de développement.....	83, 84
Installation.....	21
Licence d'utilisation.....	18
Mode super-utilisateur.....	79
Prévention du plagiat.....	79
Projets publics.....	86
Pseudonyme actif.....	80
Pseudonyme rattaché aux fichiers projet.....	81
Statistiques de projet.....	85
Pas-à-pas.....	<i>Voir Exécution pas-à-pas</i>
Pascal	
Introduction.....	17
Pédagogie	
Introduction.....	18
Période d'évaluation	
Enregistrement.....	23
Licence d'utilisation.....	19
Perl	
Constantes et variables.....	97
PI (π).....	117, 194
Pile d'appels	
Inspection de la pile d'appels.....	66
Interface de l'exécution pas-à-pas.....	63
Pile d'appels	
Configuration de la console d'exécution.....	73
PLAF.....	<i>Voir PLAFOND</i>
PLAFOND.....	117, 194
PLAN.....	<i>Voir PLANCHER</i>
PLANCHER.....	117, 195
Point d'arrêt.....	66
Couleurs pour exécution pas-à-pas.....	77
Interface de l'exécution pas-à-pas.....	63
Modes d'exécution pas-à-pas.....	64
Points d'arrêt.....	66
Surligne d'instructions en exécution pas-à-pas.....	57
Surligne de contenu.....	49
Police	
Configuration de l'éditeur graphique.....	58
Police de caractères	
Configuration des éditeurs.....	72
Polymorphe contextuel	
Constantes et variables.....	97
L'affectation.....	100
Opérations.....	98
POS.....	<i>Voir POSITION</i>
POSITION.....	118, 119, 196
Position d'élément (de conteneur).....	101
POUR.....	204
Structure POUR.....	141
Presse-papiers	
Commandes de l'éditeur textuel accessibles via les menus.....	50
Contrôle du copier-coller.....	82
Conversion d'organigramme en pseudo-code.....	86
Éditer un document textuel.....	48
Prévention du plagiat.....	79
Configuration du mode super-utilisateur et du système de mises à jour.....	75
Création du projet <i>LARP</i>	93
Débridage de l'environnement de développement.....	83
Licence d'utilisation.....	18
Mode super-utilisateur.....	79
Projets publics.....	86
Statistiques de projet.....	85
Priorité des opérateurs.....	125
Animation.....	68
Opérateurs arithmétiques.....	113
Projet public.....	86
Statistiques de projet.....	85
Pseudo-	
Navigateur de documents.....	41
Pseudo code	
Un premier algorithme.....	89
Pseudo-code.....	89
Conversion d'organigramme en pseudo-code.....	85
Couleurs dans les éditeurs.....	77
Noms de modules.....	146
Panneau de modèles.....	41
Pseudonyme	
Chiffrement des documents.....	82
Commander des clés de débridage.....	25
Configuration du mode super-utilisateur et du système de mises à jour.....	75
Création du projet <i>LARP</i>	93

Débridage de l'environnement de développement	84	Lecture et écriture	105
Mode super-utilisateur	79	Lecture via un canal d'entrées/sorties	165
Panneau de statut	45	L'instruction de lecture	106
Projets publics	86	L'instruction d'écriture	107
Pseudonyme actif	80	SÉPARATEUR	109, 205
Pseudonyme rattaché aux fichiers projet	81	Séquence d'échappement	
Sélection	80	Ouvrir un fichier	162
Statistiques de projet	85	Séquence d'échappement	100
Pseudonyme actif	80	Shareware	<i>Voir Partagiciel</i>
Contrôle du copier-coller	82	SI	
Puissance	113, <i>Voir ^</i>	Structure SI-SINON-SI	130
R			
RACINE	117, 197	Structures conditionnelles	121
ARCTANGENTE	177	Structures SI et SI-SINON	126
Rappels d'enregistrement	23	Syntaxe (structure SI)	201
Procédure d'enregistrement	24	Syntaxe (structure SI-SINON)	201
Recherche	<i>Voir POSITION</i>	Syntaxe (structure SI-SINON-SI)	202
Recherche et remplacement	48	Signet	47
Recherche et remplacement dans un organigramme	57	Commandes de l'éditeur textuel accessibles via le	
Récursivité	173	clavier	51
Configuration de la console d'exécution	73	SIN	<i>Voir SINUS</i>
Opérateurs relationnels	123	SINON	
RÉFÉRENCE		Structure de sélection	134
Déclaration des paramètres d'un module	152	Structure SI-SINON-SI	130
Modules auxiliaires avec paramètres	150	Structures SI et SI-SINON	126
Registres de Windows		Syntaxe (structure de sélection)	202
Configuration de LARP	71	Syntaxe (structure SI-SINON)	201
Répertoire		Syntaxe (structure SI-SINON-SI)	202
Ouvrir un fichier	162	SINON SI	202
RÉPÉTER		SINUS	117, 197
Structure RÉPÉTER-JUSQU'À	139	ARCTANGENTE	177
Syntaxe (structure RÉPÉTER-JUSQU'À)	203	Souris	
RÉPÉTER-JUSQU'À		Commandes d'édition de l'éditeur graphique	58
Structure POUR	141	Commandes d'édition de l'éditeur textuel	49
REQUÊTE	205	Contrôle de l'éditeur textuel via la souris	51
Console d'exécution	45	SOUSENSEMBLE	118, 119, 198
Le séparateur	109	Soustraction	<i>Voir -</i>
REQUÊTE	108	SparKey	
Restrictions de garantie	20	Messages reliés à l'environnement de	
Retour de chariot		développement	207
Le séparateur	110	Spécificateur de format	185
L'instruction de lecture	106	Statistiques de projet	85
L'instruction de requête	108	Projets publics	86
L'instruction d'écriture	107	Statut	44
Séquences d'échappement	100	Structure de sélection	134
Structures SI et SI-SINON	128	Structure POUR	<i>Voir Structures répétitives</i>
Retourner		Structure RÉPÉTER-JUSQU'À	<i>Voir Structures répétitives</i>
Modules avec valeur de retour	155	Structure SI	<i>Voir Structures conditionnelles</i>
RETOURNER	201	Structure SI-SINON	<i>Voir Structures conditionnelles</i>
Éditer un organigramme	53	Structure SI-SINON-SI	<i>Voir Structures conditionnelles</i>
Modules auxiliaires	147	Structure TANTQUE	<i>Voir Structures répétitives</i>
Modules avec valeur de retour	156	Structure RÉPÉTER-JUSQU'À	139
S			
Sauvegarde de projet	96	Structures conditionnelles	121
Sauvegarde de sécurité	70	Détection de fin de contenu via un canal	
Sauvegarde de sécurité	70	d'entrées/sorties	166
Configuration de la console d'exécution	74	Structure de sélection	134
Sélection des couleurs		Structure SI-SINON-SI	130
Console d'exécution	46	Structures SI et SI-SINON	126
Éditer un document textuel	48	Structures SI-SINON imbriquées	128
Surligne de contenu	49	Structures imbriquées	
SÉLECTIONNER	202	Structure SI-SINON-SI	130
Structure de sélection	134	Structures SI-SINON imbriquées	128
Séparateur		Structures itératives	<i>Voir Structures répétitives</i>
Écriture via un canal d'entrées/sorties	166	Structures répétitives	137
		Détection de fin de contenu via un canal	
		d'entrées/sorties	166
		Récursivité	173
		Structure POUR	141
		Structure RÉPÉTER-JUSQU'À	139
		Structure TANTQUE	137

